CHAPTER

17

Statements

SAS Statements under OpenVMS

A SAS statement is a directive to the SAS System that either requests that SAS perform a certain operation or provides information to the system that might be necessary for later operations.

All SAS statements are completely described in *SAS Language Reference: Dictionary*. Those that are described here have syntax and usage that is specific to the OpenVMS operating environment.

ABORT

Stops executing the current DATA step, SAS job, or SAS session

Language element: statement

Valid: in a DATA step

OpenVMS specifics: action of ABEND and RETURN; maximum value of *n*

Syntax

ABORT <ABEND | RETURN> <*n*>;

Note: This is a simplified explanation of the ABORT statement syntax. For the complete explanation, see *SAS Language Reference: Dictionary*. △

no argument
stops processing the current DATA step. Additional actions and results depend on the method of operation.

ABEND
causes abnormal termination of the current SAS job or session. Results depend on the method of operation.

RETURN
causes the immediate normal termination of the current SAS job or session. Results depend on the method of operation.

n
is an integer value that enables you to specify a condition code that SAS returns to OpenVMS when it stops executing. The value of *n* can range from –2,147,483,648 to 2,147,483,647.

Details If you specify ABORT ABEND, the symbol SAS$STATUS is set to 999. If you specify ABORT RETURN, the symbol SAS$STATUS is set to 12. Both the ABEND and RETURN arguments terminate the SAS job or session.

The value of *n* can range from –2,147,483,648 to 2,147,483,647.

See Also

- ABORT statement in *SAS Language Reference: Dictionary*
- "Determining the Completion Status of a SAS Job" on page 43

ATTRIB

Associates a format, informat, label, or length or all four with one or more variables

Language element: statement
Valid: in a DATA step
OpenVMS specifics: length specification

Syntax

ATTRIB *variable-list-1 attribute-list-1*
< . . . *variable-list-n attribute-list-n*>;

variable-list
names the variables that you want to associate with the attributes.

attribute-list
specifies one or more attributes to assign to *variable-list*. Specify one or more of these attributes in the ATTRIB statement:

FORMAT=*format*
associates a format with variables in *variable-list*.

INFORMAT=*informat*
associates an informat with variables in *variable-list*.

LABEL='*label*'
associates a label with variables in *variable-list*.

LENGTH=<$>*length*
specifies the length of variables in *variable-list*. A dollar sign ($) is required in front of the length of character variables. For character variables, the value of *length* is 1 to 32,767.

Details Under OpenVMS Alpha, the minimum length that you can specify for a numeric variable is 3 bytes.

Under OpenVMS VAX, the minimum length that you can specify for a numeric variable is 2 bytes.

See Also

- ATTRIB statement in *SAS Language Reference: Dictionary*

FILE

Specifies the current output file for PUT statements

Language element: statement

Valid: in a DATA step

OpenVMS specifics: valid values for *file-specification*; valid values for *host-option-list*

Syntax

FILE *file-specification* <*option-list*>
<*host-option-list*>;

file-specification
can be any type of file specification discussed in "Identifying External Files to the SAS System" on page 166.

option-list
specifies portable options for the FILE statement. For information about these options, see the FILE statement in *SAS Language Reference: Dictionary*.

host-option-list
specifies external I/O statement options that are specific to the OpenVMS environment. These options can be any of the following:

ALQ=

CC=

DEQ=

FAC=

GSFCC=

KEY=

KEYVALUE=

LINESIZE=

LRECL=

MBC=

MBF=

MOD

NEW

OLD

PAGESIZE=

RECFM=

SHR=

For information about these options, see “Host-Specific External I/O Statement Options” on page 361 in the FILENAME statement.

Many of the DCL print qualifiers are also supported as host options in the FILE and FILENAME statements. For details, see “Printer Options in the FILENAME and FILE Statements” on page 371 in the FILENAME statement.

You can intersperse options from *option-list* and *host-option-list* in any order.

Note: When using the PIPE device with the FILE statement, only the LRECL host option is supported. △

Details By default, PUT statement output is written to the SAS log. Use the FILE statement to route this output to either the same external file to which procedure output is written or to a different external file. You can indicate whether or not carriage control characters should be added to the file.

You can use the FILE statement in conditional (IF-THEN) processing because it is executable. You can also use multiple FILE statements to write to more than one external file in a single DATA step.

Example

The following is an example of a FILE statement:

```
file prices;
```

This FILE statement uses the default filename form of the file specification (PRICES has not been assigned as a SAS fileref or OpenVMS logical name). Therefore, the SAS System looks for the file PRICES.DAT in the current directory.

When the SAS System writes a file, it creates a new version by default. For example, if your default directory contains versions 1 and 2 of the file PRICES.DAT, then this FILE statement writes PRICES.DAT;3 in your default directory.

If you want to append output lines to the most recent version of an external file, use the MOD option in the FILE statement. For instance, from the previous example your default directory contains three versions of PRICES.DAT. The following statement appends data lines to PRICES.DAT;3:

```
file prices mod;
```

To reference an explicit version of a file, use the version number as part of the file specification within a quoted string. For example, the following FILE statement writes to version 1 of the file:

```
file 'prices.dat;1';
```

See Also

- FILE statement in *SAS Language Reference: Dictionary*
- Statement: "FILENAME" on page 357
- "Identifying External Files to the SAS System" on page 166
- Command: "FILE" on page 228

FILENAME

Associates a SAS fileref with an external file

Language element: statement

Valid: anywhere in a SAS program

OpenVMS specifics: valid values for *device-type*; valid values for *external-file*; valid values for *host-option-list*

Syntax

FILENAME *fileref* <*device-type*>
'*external-file*' <*host-option-list*>;

FILENAME *fileref device-type* <'*external-file*'>
<*host-option-list*>;

Note: This is a simplified version of the FILENAME statement syntax. For the complete syntax and its explanation, see the FILENAME statement in *SAS Language Reference: Dictionary*. △

fileref
is any valid fileref and can be up to eight characters long. The first character must be a letter (A to Z), an underscore (_), a dollar sign ($), a pound sign (#), or an at sign (@). Subsequent characters can be any of these characters, or they can be numbers. Neither OpenVMS nor the SAS System distinguishes between uppercase and lowercase letters in filerefs or in filename specifications. This is a required argument. See "Reading from and Writing to OpenVMS Commands (Pipes)" on page 179 for

information on assigning a fileref to a pipe to read from and write to OpenVMS commands.

The following are some examples of valid filerefs:

- TEST_1
- MYFILE
- abc123

The following are some examples of invalid filerefs:

- ALONGFILENAME (longer than eight characters)
- 123_test (begins with a number)
- TEST%X (contains an invalid character (%)).

device-type

specifies an output device. For details about device types, see "Device-Type Keywords" on page 360. The device-type keyword must follow *fileref* and must precede *external-file* (if an external file is used).

'external-file'

can be any valid file specification in quotation marks. The file specification must be a valid OpenVMS pathname to the external file that you want to access; therefore, the level of specification depends on your location in the directory structure. The number of characters in the quoted string must not exceed the maximum filename length that OpenVMS allows (255 characters).

Under OpenVMS, you can specify concatenations of files when reading and writing external files from within the SAS System. Concatenated files consist of two or more file specifications, enclosed in quotation marks and separated by commas. The following is an example of a valid concatenation specification:

```
filename alldata 'test.data1, test.data2,
                  test.data3';
```

For a complete discussion, see "Using OpenVMS Pathnames to Identify External Files" on page 168

For Version 8, if you specify a version number for the file in a FILENAME statement, the version number of the file is not increased. For example, the following FILENAME statement will produce only one file, **test.dat;1**:

```
filename myfile 'test.dat;1';
   data;
   file myfile;
   put 'hello';
   run;
   data;
   file myfile;
   put 'hello again';
   run;
```

For more details, see "Using OpenVMS Pathnames to Identify External Files" on page 168. For more information about valid OpenVMS pathnames, refer to *OpenVMS User's Manual*.

host-option-list

names any of the following external I/O statement options:

ALQ=

CC=

DEQ=

FAC=

GSFCC=

KEY=

KEYVALUE=

LINESIZE=

LRECL=

MBC=

MBF=

MOD

NEW

OLD

PAGESIZE=

RECFM=

SHR=

These options control how the external file is processed and are specific to the OpenVMS environment. For information about these options, see "Host-Specific External I/O Statement Options" on page 361.

Many of the DCL print qualifiers are also supported as host options in the FILENAME and FILE statements. For details, see "Printer Options in the FILENAME and FILE Statements" on page 371.

Note: When using the PIPE device, only the LRECL= host option is supported. △

Details The FILENAME statement is significantly different from the LIBNAME statement. The FILENAME statement is for external files only and references a specific filename. The LIBNAME statement is for SAS files only, and it generally specifies directory- and subdirectory-level information only (except when you are assigning a libref for use with the XPORT, OSIRIS, or SPSS engines). Also, unlike a libref, you can associate a fileref with a file that does not yet exist; when you use the fileref in a FILE statement or command, the file is created according to your specifications.

You can choose to use only a directory name in the FILENAME statement (the directory must exist, except when doing a concatenation). You must then use the fileref and the filename in subsequent statements as discussed in "Using Aggregate Syntax to Identify External Files" on page 169. The SAS System supplies a default file type.

Reserved Filerefs Under OpenVMS, the following are reserved filerefs (the items in parentheses are the SAS statements to which each applies):

DATALINES (INFILE)
: specifies that input data immediately follow a DATALINES statement in your SAS stream. The only time you need to use the INFILE DATALINES fileref is when you want to use INFILE statement options to read in stream data.

LOG (FILE)
: specifies that output lines produced by PUT statements are written to the SAS log. LOG is the default destination for output lines.

PRINT (FILE)
: specifies that output lines produced by PUT statements are written to the procedure output file, the external file to which SAS procedure output is written.

Device-Type Keywords When you specify a *device-type* in a FILENAME statement, the *external-file* argument is optional. If you do specify an external file, its meaning depends on which device type you specified. For example, the following SAS program sends the output file to the printer that is associated with the SYS$PRINT queue:

```
filename myfile printer;

data test;
   file myfile;
   put 'This is spooled to a printer.';
run;
```

The following are the valid device-type keywords and their meanings:

DISK
sends the output to or reads the input from a disk device. This is the default behavior if you do not specify any device-type keywords in the FILENAME statement. You must specify an external file with this keyword.

TERMINAL
sends the output to or reads the input from a terminal device. If you do not specify an external file, the output goes to the SYS$OUTPUT output stream. If you do specify an external file, the file specification is ignored and SYS$OUTPUT is still used.

If you want to display and enter data in the same step, then issue two FILENAME statements and use one fileref for input and one for output.

PRINTER
spools the output to a printer queue. If you do not specify an external file, the output is sent first to a temporary file on disk and then to the SYS$PRINT queue. Then the disk file is deleted. If you do specify a file, the output is sent to that file and then sent to the SYS$PRINT queue. In this case, the disk file is not deleted. To send the output to a printer queue other than SYS$PRINT, use the QUEUE= option in the FILENAME or FILE statement. For information about printer options, see "Printer Options in the FILENAME and FILE Statements" on page 371.

PLOTTER
spools the output to a printer queue that has been assigned to a plotter. This keyword works the same as the PRINTER keyword except that the file format is valid for a plotter device. The file that is created is an OpenVMS print file; it has a variable record format, with a 2-byte, fixed-length control field.

For every record, the 2-byte control field is set to NULL, indicating no carriage control.

TAPE
sends the output to or reads the input from a tape device. You are responsible for allocating the tape drive and mounting the tape before output is sent to the device. If you do not specify an external file, the output is sent to the tape device that is associated with the logical name SASTAPE. If you do specify a file, the device portion of the filename must be a valid tape device.

TEMP
is a temporary file that can only be accessed through the logical name and is only available while the logical name exists. If a physical pathname is specified, an error is returned. Files manipulated by the TEMP device can have the same attributes and behave identically to DISK files.

DUMMY
sends the output to NLA0: (the null device). If you specify an external file, the file specification is ignored. This device-type keyword is useful when you are debugging SAS programs. You can test your algorithms without actually writing output files.

PIPE
sends the output to or reads the input from an OpenVMS command. For more information, see "Reading from and Writing to OpenVMS Commands (Pipes)" on page 179.

These keywords are valid only in the FILENAME statement. However, a fileref for which you specified a device-type keyword can be used in the SAS windowing environment commands and in the FILE, INFILE, and %INCLUDE statements. In order to use these devices correctly, you must specify the device-type keyword in the FILENAME statement. (If you use a device specification only in the quoted file specification of a FILE or INFILE statement, the results are unpredictable.) For example, to correctly send output to the display, use the following statements:

```
filename myfile terminal;

data test;
   file myfile;
   put 'This is my test';
run;
```

By contrast, the following lines are incorrect and may yield unpredictable results:

```
data test2;
   file 'sys$output';
   put 'This may not work in all cases.';
run;
```

When you use the TERMINAL device type with the INFILE statement, you terminate input by pressing CTRL-Z.

Host-Specific External I/O Statement Options The following external I/O statement options can be used in the FILE, INFILE, and FILENAME statements. Note that some of these options, such as ALQ=, have the same names as SAS data set options. Do not confuse the two types of options. You cannot use data set options with external files.

This list includes only options that are specific to the OpenVMS environment. For a complete list of external I/O statement options, see "Summary of External I/O Statement Options" on page 370 and the statements chapter in *SAS Language Reference: Dictionary*.

The following descriptions include an explanation of the option, its valid and default values, and whether it is used for input, output, or both.

If the same option is used in both the FILENAME and FILE statements or in both the FILENAME and INFILE statements, the FILE or INFILE value takes precedence over the value used in the FILENAME statement.

ALQ=
specifies the number of blocks initially allocated to an external file when it is created. The value can range from 0 to 2,147,483,647. If the value is 0 (the default), the minimum number of blocks required for the given file format is used.

The ALQ= option (allocation quantity) is used for output and corresponds to the FAB$L_ALQ field in OpenVMS Record Management Services (RMS). For additional details, refer to *Guide to OpenVMS File Applications*.

BLKSIZE= | BLK=
is no longer supported in the OpenVMS operating environment.

CC=
specifies the carriage-control format of the SAS log and listing file. This option has three possible values:

FORTRAN
indicates FORTRAN carriage-control format. This is the default for print files.

PRINT
indicates OpenVMS print format.

CR
indicates OpenVMS carriage-return, carriage-control format. This is the default for nonprinting files.

Only SAS print files are affected by the CC= option. The CC= option is used for output.

The CC= option also exists as a SAS system option (see "CC=" on page 403). If you specify this option both as a system option and in the FILENAME or FILE statement, then SAS uses the value that you specified in the FILENAME or FILE statement.

DEQ=
specifies the number of blocks added when OpenVMS RMS automatically extends an external file during a write operation. The value can range from 0 to 65,535. The default value is 0, telling OpenVMS RMS to use the process's default value. A large value results in fewer file extensions over the life of the file; a small value results in numerous file extensions over the life of the file. A file with numerous file extensions may be noncontiguous, thereby slowing record access.

The DEQ= option (default file extension quantity) is used for output and corresponds to the FAB$W_DEQ field in OpenVMS RMS. For additional details, see *Guide to OpenVMS File Applications*.

FAC=
overrides the default file access attributes used for external files. Use this option to indicate the level of access you want to allow for an external file. You can allow read, write, update, and delete access (as well as no access). By default with external files, files opened for input allow read access, files opened for output allow write access, and files opened for update allow read and write access. The form of the FAC= option is

```
FAC=access-option-list
```

where *access-option-list* can be one of the following:

DEL	specifies delete access.
GET	specifies read access.
PUT	specifies write access.
UPD	specifies update access.

You can combine these values in any order. For example, specifying the following indicates you want delete, read, and write access:

```
fac=(del,get,put)
```

By also specifying the SHR= option, you can allow other users concurrent access to an external file, either through a separate SAS session or with another application. To allow sharing, you must include the values for FAC= in the list of values for SHR= (but the reverse is not true). For more information, see the description of the SHR= option later in this section.

The FAC= option (file access) can be used for both input and output and corresponds to the FAB$B_FAC field in OpenVMS RMS or the ACCESS attribute

when using File Definition Language (FDL). For additional details on file sharing, see *Guide to OpenVMS File Applications*.

GSFCC=
specifies the file format of graphic stream files (GSF files). When specified on the FILENAME statement, it affects only the GSF files created using that fileref. The accepted values are

PRINT — creates a GSF file. It is a VFC format file with carriage control set to null. These files can be used with most utilities with the exception of some file transfer protocols, such as Kermit. This is the default value for this option.

CR — creates a carriage return carriage control file.

NONE — creates a file with no carriage control. This format is useful if you plan to download the file to a personal computer.

KEY=
specifies which key the SAS System uses to read the records in an RMS file with indexed organization. The KEY= option is always used with the KEYVALUE= option. For details, see “Using the KEY= Option” on page 366 and “Using the KEYVALUE= Option” on page 367.

KEYVALUE=
specifies the key value with which to begin reading an indexed file. For details, see “Using the KEYVALUE= Option” on page 367.

LINESIZE=
specifies the line size for input or output. The value can range from 10 to 32,767. The default is 80 for interactive jobs (interactive line mode and the SAS windowing environment) and 132 for noninteractive and batch jobs for print files.

This option also exists as a SAS system option (see “LINESIZE=” on page 418). If this option is used both as a system option and in the INFILE or FILE statement, the SAS System uses the value given in the INFILE or FILE statement.

LRECL=
specifies the record length of the output file. If you do not specify a record length, the default is varying length records. For input, the existing record length is used by default. If the LRECL= option is used, the input records are padded or truncated to the specified length.

The maximum record size for OpenVMS is 32,767. LRECL values greater than 32,767 are valid only when reading and writing to tape. If an LRECL value greater than 32,767 is specified when writing to a non-tape device, the LRECL value is set 32,767. You should use the maximum LRECL values for the various file types provided in Table 17.1 on page 364.

Because the FLOWOVER option on the FILE statement is the default, lines that are longer than the length specified by the LRECL= option are split.

When accessing unlabeled tapes, you must use LRECL=. The minimum value in this case is 14. For more information, see “Reading from an Unlabeled Tape” on page 177.

The LRECL= option is used for both input and output.

Table 17.1 Maximum LRECL Values for File Types

File Organization	Record Format	Maximum LRECL Value
Sequential	Fixed length	32,767
Sequential (disk)	Variable length	32,765
Sequential (disk)	VFC	32,767-FSZ
Sequential (disk)	Stream	32,767
Sequential (disk)	Stream-CR	32,767
Sequential (disk)	Stream-LF	32,767
Sequential (ANSI Tape)	Variable length	9,995
Sequential (ANSI Tape)	VFC	9,995-FSZ
Relative	Fixed length	32,255
Relative	Variable length	32,253
Relative	VFC	32,253-FSZ
Indexed, Prolog 1 or 2	Fixed length	32,234
Indexed, Prolog 1 or 2	Variable length	32,232
Indexed, Prolog 3	Fixed length	32,224
Indexed, Prolog 3	Variable length	32,224

FSZ represents the size, in bytes, of the fixed control area in a record with VFC record format.

MBC=
specifies the size of the I/O buffers that OpenVMS RMS allocates for a particular file. The value can range from 0 to 127 and represents the number of blocks used for each buffer. By default, this option is set to 0 and the default values for the process are used.

The MBC= option (multiblock count) is used for both input and output to control the allocation for a particular file. If you want to control the allocation size for all the external files used during the current SAS session, you can use the MBC= option in every FILE, FILENAME, or INFILE statement. You can also use the DCL SET RMS_DEFAULT command to specify a process default, and let the SAS System value default to the process's default values.

The MBC= option corresponds to the RAB$B_MBC field in OpenVMS RMS or the CONNECT MULTIBLOCK_COUNT attribute when using FDL. This option is not supported for DECnet operations. For additional details, see *Guide to OpenVMS File Applications*.

MBF=
specifies the number of I/O buffers you want OpenVMS RMS to allocate for a particular file. The value can range from 0 to 127 and represents the number of buffers used. By default, this option is set to 2 buffers. If a value of 0 is specified, the default value for the process is used.

The MBF= option (multibuffer count) is used for both input and output to control the number of buffers allocated for a particular file. If you want to control the number of buffers allocated for all the external files used during the SAS session, you can use the MBF= option in every FILE, FILENAME, or INFILE statement.

The DCL SET RMS_DEFAULT command can be used to specify a process default. Then, you can let the SAS System value default to the process's default values.

The MBF= option corresponds to the RAB$B_MBF field in OpenVMS RMS or the CONNECT MULTIBUFFER_COUNT attribute when using FDL. This option is not supported for DECnet operations. For additional details, see *Guide to OpenVMS File Applications*.

MOD
opens the file referenced for append. This option does not take a value. An existing file of the name given in the FILENAME or FILE statement is opened and new data appended to the end.

NEW
opens a new file for output. This option does not take a value. This is the default action for files referenced by a FILE statement. Under OpenVMS, this option is synonymous with the OLD option.

OLD
opens a new file for output. This option does not take a value. This is the default action for files referenced by a FILE statement. Under OpenVMS, this option is synonymous with the NEW option.

PAGESIZE=
specifies the page size for output. The default is the display setting for interactive jobs (interactive line mode and the SAS windowing environment) and 60 for noninteractive and batch jobs. The value can range from 15 to 32767.

This option also exists as a SAS system option (see "PAGESIZE=" on page 429). If this option is used both as a system option and in the FILE statement, the SAS System uses the value given in the FILE statement.

RECFM=
specifies the record format of the output file. Values for the RECFM= option are

F specifies fixed length.

V specifies variable length.

D specifies you are accessing unlabeled tapes with the PUT and INPUT DATA step statements. For more information, see "Reading from an Unlabeled Tape" on page 177.

If the RECFM= option is not used, the value defaults to V for output files. For input files, the default value is the record format of the file.

This option is used for both input and output.

SHR=
overrides the default file-sharing attributes used for external files. With this option, you can indicate the access level you want to give other users. You can allow read, write, update, and delete access (as well as no access). By default with external files, files opened for input allow shared read access, and files opened for output or update do not allow shared access.

However, you can allow other users to have read and write access to a file that you are opening for input only. To accomplish this, use the SHR= option. The syntax of the SHR= option is

`SHR=`*`share-option-list`*

where *share-option-list* can be one of the following:

DEL specifies delete access.

GET specifies shared read access.

NONE specifies no shared access.

PUT specifies shared write access.

UPD specifies update access.

You can combine these values in any order. For example, specifying the following indicates you want shared delete, read, and write access:

```
shr=(del,get,put)
```

To allow shared access, the values for FAC= must be included in the list of values for SHR= (but the reverse is not true).

This option corresponds to the FAB$B_SHR field in OpenVMS RMS or the SHARING attribute when using FDL. For more information about file sharing, see *Guide to OpenVMS File Applications*.

The SHR= option is used for both input and output.

Note: When using the PIPE device, only the LRECL= host option is supported. △

Using the KEY= Option The KEY= option is used for input. It is always used with the KEYVALUE= option. A key is a record field that identifies the record to help you retrieve the record in an indexed file. The two types of keys are primary and alternate. Data records are stored in the file in the order of their primary key. Alternate keys (also called secondary keys) create alternate indexes in the file. The alternate index can then be used to process the records in order of the alternate key. The only difference between the primary key and the alternate key is that the records are actually stored in the order of the primary key, whereas the alternate key provides a means of accessing them.

The key number is zero-based, so KEY=0 (the default) specifies that the records be read in sorted order by the primary key. KEY=1 specifies the use of the first secondary key to access the records.

To use the SAS System to write to an indexed file, you can either create an empty indexed file or use any existing indexed file. If you create an empty indexed file, use FDL to specify the file characteristics, including the type and location of primary and secondary keys. (For more information on FDL, see *OpenVMS File Definition Language Facility Manual*.) The following is an example program:

```
/*-----------------------------------------*/
/* This SAS program accesses an empty      */
/* indexed file that has been previously   */
/* created. The data are appended to the   */
/* file. Primary key #0 is of type character */
/* and is in bytes 0-2. Secondary key #1 is  */
/* of type character and is in bytes 3-5.    */
/*-----------------------------------------*/
filename myfile 'indexed.dat';
   /* Load the indexed file, primary key in */
   /* sorted order.                        */
data _null_;
   file myfile mod;
   put 'aaaccc';
   put 'bbbaaa';
run;
   /* Print out in primary key sorted order. */
data _null_;
      /* Key=0 is the default. */
   infile myfile;
   input first $3. second $3.;
```

```
   put first= second=;
run;
```

This program produces the following output:

```
first=aaa second=ccc
first=bbb second=aaa
```

In contrast, consider setting KEY=1 as in the following example:

```
/* Print out in secondary key sorted order. */
data _null_;
   infile myfile key=1;
   input first $3. second $3.;
   put first= second=;
run;
```

This program produces the following output:

```
first=bbb second=aaa
first=aaa second=ccc
```

All keys are defined in RMS when the file is created. For more information about defining and using keys in an indexed file, see *Guide to OpenVMS File Applications*.

Using the KEYVALUE= Option The KEYVALUE= option is always used with the KEY= option, which specifies the key used by the SAS System to read the records in an RMS file with indexed organization. When you use the KEYVALUE= option, the file is input sequentially, beginning with the value you specified. It is similar to the FIRSTOBS= option used with a sequential-format file. You can specify a SAS variable name with the KEYVALUE= option to drive random reads from the file. The KEYVALUE= option is used for input.

Valid forms of the KEYVALUE= option are

```
KEYVALUE operator value
   AND KEYVALUE operator value

KEYVALUE=SAS-variable
```

where *operator* can be one of the following:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to.

where *value* can be one of the following data types:

- integer
- quoted string
- quadword (signed or unsigned)
- packed decimal
- date/time.

The key specified in the KEY= option is used with the KEYVALUE= option. The defined order of the key specified must match the direction of the operator given in the KEYVALUE= option. For example, if the key is an ascending order key, the < and <= operators are invalid operators. When the value of KEYVALUE= is a constant value, the

file is processed sequentially by key, beginning with the given value. When the value of KEYVALUE= is a SAS variable, the first record with a key satisfying the criterion of the KEYVALUE expression is read from the file. Note that the SAS variable value must match the key value of one of the records exactly or an end-of-file condition occurs.

The data type of the key specified in the KEY= option must also correspond to the type given as the value for the KEYVALUE= option. The following RMS data types are supported by the KEYVALUE= option:

- unsigned 2-byte binary
- unsigned 4-byte binary
- unsigned 8-byte binary
- signed 2-byte binary
- signed 4-byte binary
- signed 8-byte binary
- left-justified string of characters
- packed decimal string of characters.

The SAS System converts an integer value to the correct format of the supported numeric types. Character string values are not changed when used for the character type.

When you use date-time values, the data are stored in signed 8-byte binary RMS key fields. Use the VMSTIMEF. format to convert SAS date-time values to signed 8-byte binary values. When you access records through a date-time value key using the KEYVALUE=*SAS-variable* option, the SAS variable must have one of the following SAS formats or informats associated with it:

DATE*w*.

DATETIME*w*.

DDMMYY*w*.

JULIAN*w*.

MMDDYY*w*.

MONYY*w*.

YYMMDD*w*.

YYQ*w*.

A format or informat must be associated with the SAS variable because neither the variable value nor the field value within the record indicates that the data represent date-time values. For more information about these formats and informats, refer to *SAS Language Reference: Dictionary*.

Suppose you want to input an indexed file that has an alternate key defined as a signed 4-byte integer in descending sort order. You can process only the records with the values less than 5,000 with the following DATA step:

```
filename in 'indexed.dat' key=2 keyvalue<5000;
data _null_;
   infile in;
   input name $9. num;
   put name num;
run;
```

Using Compound Expressions You may further restrict the number of records read by using a compound KEYVALUE expression. For example, suppose you want to input an indexed file that has a primary key defined as a signed 2-byte integer in ascending sort order. You can retrieve records with key values between –10 and 10 inclusive with the following FILENAME statement:

```
filename in 'indexed.dat' keyvalue>=-10 and
   keyvalue<=10;
```

When given a compound KEYVALUE expression, the SAS System reads records from the input file until a record is read with a key exceeding the upper boundary, which is 10 in this example, or until the end of file is reached. Note that the AND construct has an associative property; the order of the KEYVALUE options can be reversed and the meaning preserved. However, the operators still must match the key sort order, so the following DATA step using the same indexed file described in the earlier example generates an error:

```
data wrong;
   infile 'indexed.dat' keyvalue<=-10 and
      keyvalue>=10;
   input num name $9.;
   put name num;
run;
```

This DATA step generates the following error and warning messages:

```
ERROR: Specified key on indexed file is
       defined as ascending but <, <= or
       = was used in KEYVALUE option.
NOTE: The SAS System stopped processing
      this step because of errors.

WARNING: The data set WORK.WRONG may be
         incomplete.  When this step was
         stopped there were 0 observations
         and 1 variables.
```

Using SAS Variables Using a SAS variable name as the value of the KEYVALUE= option enables you to randomly access records in the indexed file. In the previous examples of using the KEYVALUE= option, the input file was sequentially accessed. You can use any SAS variable with the KEYVALUE= option that matches the type of the key in question. When the SAS System reads from the file, it reads the first record with the key value that matches the value of the SAS variable.

For example, suppose you have a SAS data set named SALES that has three variables: SALESREP, ITEMNO, and QUANTITY. This data set contains the number of items each salesperson sold during the last month. You also have an indexed file keyed by the item numbers of the products the company sells. Stored in each record is the price of the item. Using these two files, the SAS System can easily generate a report of the revenue generated by each salesperson:

```
filename parts 'inventory.idx' key=0;
filename report 'revenue.lis';
data revenue;
   set sales;
   infile parts keyvalue=itemno;
   input itemno price;
   revenue=quantity*price;
   output @5 salesrep @30 itemno
          @50 revenue dollar3.2;
   stop;
run;
```

This sample program match-merges the observations in SALES with the records in the indexed file by item number to produce the reports. A KEYVALUE= option with a

SAS variable name can be used only with the equal sign (=) operator and cannot be used in compound KEYVALUE expressions.

Note that in the previous example, the DATA step is driven entirely by the SET statement. The DATA step terminates when all records from the data set SALES have been processed. It is possible to use the SAS variable form of the KEYVALUE= option with other types of control mechanisms. In the following example, an iterative DO loop determines the set of records read from an indexed file:

```
data example2;
   do i=1 to 20 by 2;
      infile myfile key=0 keyvalue=i;
      input var1 var2 var3 ...;
      /* .... variable processing ... */

      output var1 var2 var3 ...;
   end;
   stop;
run;
```

In this example, the DO loop is used to read every other record from MYFILE. Note that the STOP statement terminates the DATA step and closes the input file. Because the KEYVALUE=I option reads only those records specified in the DO statement, the SAS System cannot read an end-of-file indicator as it would if it were reading the file sequentially. Without the STOP statement to end the DATA step, the SAS System can get into an infinite loop by accessing the same index file repeatedly.

For more information about indexed files and keys, refer to *Guide to OpenVMS File Applications*.

Summary of External I/O Statement Options The following table lists alphabetically all available external I/O statement options, including both portable options and options that are specific to the OpenVMS environment. The *Use* column indicates whether the option is used for input, output, or both. The support of the option in the FILENAME statement is host-specific. Options that are used with the FILENAME statement are not documented in *SAS Language Reference: Dictionary*.

Table 17.2 Summary of External I/O Statement Options

Option	Use	Option	Use
ALQ= **	output	LINESIZE= *	input, output
CC= **	output	LRECL= *	input, output
COLUMN= ***	input, output	MBC= **	input, output
DELIMITER= ***	input	MBF= **	input, output
DEQ= **	output	MISSOVER ***	input
DROPOVER ***	output	MOD *	output
END= ***	input	N= ***	input, output
EOF= ***	input	NEW **	output
EXPANDTABS ***	input	NOTITLES ***	output
FAC= **	input, output	OBS= ***	input
FILENAME= ***	input, output	OLD *	output

Option	Use	Option	Use
FILEVAR ***	input, output	PAD ***	input, output
FIRSTOBS= ***	input	PAGESIZE= *	output
FLOWOVER ***	input, output	PRINT ***	input, output
GSFCC **	output	RECFM= *	input, output
HEADER= ***	output	SHR= **	input, output
KEY= **	input	SHAREBUFFERS ***	input
KEYVALUE= **	input	START= ***	input
LENGTH= ***	input	STOPOVER ***	input, output
LINE= ***	output	UNBUFFERED ***	input

* This option is also documented in *SAS Language Reference: Dictionary*.
** All the information for this option is contained in this document.
***This option is completely documented in *SAS Language Reference: Dictionary*.

Printer Options in the FILENAME and FILE Statements Many of the DCL print qualifiers are supported as host options in the FILE and FILENAME statements. If the same option is used in both the FILE and FILENAME statements, the FILE statement value overrides the FILENAME statement value. You send a file to a printer by using the PRINTER or PLOTTER device-type keyword in the FILENAME statement.

A complete list of supported options follows. For more information about the meanings of specific options, refer to *OpenVMS DCL Dictionary*.

AFTER="*quoted-string*"
specifies a time after which the file can be printed. The time can be specified as absolute time or a combination of absolute and delta times and must be enclosed in double quotation marks.

BURST=<ALL | NO>
specifies a burst page or not. The default value is NO.

CHAR=(,,,)
lists characteristics for the printer. The list can be one item or a group of items enclosed by parentheses. No spaces are allowed in the list.

COPIES=*n*
specifies the number of copies to print. The default value is 1.

FEED=<YES | NO>
specifies whether to perform a form feed at the end of the page. The default value is YES.

FLAG=<ALL | NO>
specifies whether to print a flag page preceding each file. The default value is NO.

FORM=*type*
defines the form name or number used.

HDR=<YES | NO>
controls whether a header line is printed at the top of each page. The default value is NO.

NAME=*quoted-string*
specifies the name of the submitted job shown when you issue a SHOW QUEUE command. The default is the filename. The *quoted-string* argument can contain spaces.

NOTE=*quoted-string*
specifies a message to appear on the flag page. The *quoted-string* argument can contain spaces.

NOTIFY=<YES | NO>
controls whether to notify the user when the job is finished. The default value is NO.

PARAM=<">(,,,)<">
sends a list of up to eight parameters to the printer device. The PARAM= value can be one item without parentheses, or a group of items enclosed by parentheses. If the value contains blanks or nonalphanumeric characters, enclose the entire value argument in single or double quotation marks.

PASSALL=<YES | NO>
specifies whether all formatting is bypassed and sent to the device driver with formatting suppressed. The default value is NO.

QUEUE=<">*queue-name*<">
specifies the name of the printer queue to send the job to. If this option is not used, the job is submitted to the SYS$PRINT queue. If the queue name contains characters not recognized by the SAS System, it must be enclosed in single quotation marks; for example, SYS$PRINT must be enclosed in quotation marks, but CLXYJ31 does not need to be. The *queue-name* argument cannot contain any spaces.

RESTART=<YES | NO>
restarts the job after a crash. The default value is YES.

SETUP=(,,,)
sets up module names to extract from the device control library. The list can be a single item or a group of items enclosed by parentheses.

SPACE=<1 | 2>
specifies double- or single-spacing. The default value is single.

TRAILER=<ALL | NO>
prints a trailer page at the end of the file. The default value is NO.

Examples

Example 1: Associating a Fileref with an External File In this example, the FILENAME statement associates the fileref PGMSAS with an external file that contains a SAS program. PGMSAS is then used as the fileref in the %INCLUDE statement to read a file that contains SAS statements.

```
filename pgmsas '[yourdir]prog1.sas';
%include pgmsas;
```

Example 2: Using a File as Input to an INFILE Statement Consider the following FILENAME statement:

```
filename myfile '[mydir]';
```

If you want to use a file in [MYDIR] named SCORES02.DAT as input to an INFILE statement, issue the following statement:

```
infile myfile(scores02);
```

The SAS System assumes a file type of .DAT in the INFILE statement.

If you do not specify a file type in the external file specification, the default file type is .DAT. For example, the following FILENAME statement associates the fileref MYFILE with a file named SURVEY.DAT:

```
filename myfile 'survey';
```

Example 3: Using Printer Options The following statement sends a copy of the file A.LIS to the CLXYJ31 queue, holds it until 2:00 p.m., and then prints two copies:

```
filename x printer 'a.lis' queue=clxyj31
                           after="14:00:00" copies=2;
```

The following statement creates the file A.LIS but does not send it to the printer because the PRINTER device-type keyword is not used. The AFTER= option is ignored.

```
filename x 'a.lis' after="14:00:00";
```

The following statement sends the file A.LIS to the SYS$PRINT queue, holding it until 2:30 p.m.:

```
filename x printer 'a.lis' after="14:30:00";
```

The following statement creates a temporary file called SAS0000*n* and sends it to the SYS$PRINT queue. The file is deleted after printing.

```
filename x printer;
```

The following statement creates the file CLXYJ31.DAT and sends it to the SYS$PRINT queue. The file is not deleted after printing.

```
filename x printer 'clxyj31';
```

As a final example, the following lines create a file A.LIS and send it to the SYS$PRINT queue. The job name submitted is MYFILE.

```
filename x printer 'a.lis';
data a;
   file x name="myfile";
   . . . more SAS statements . . .
run;
```

See Also

- FILENAME statement in *SAS Language Reference: Dictionary*
- Chapter 7, "Using External Files and Devices," on page 165

FOOTNOTE

Prints up to ten lines of text at the bottom of the procedure output

Language element: statement
Valid: anywhere in a SAS program
OpenVMS specifics: maximum length of footnote

Syntax

FOOTNOTE <*n*> <'*text*' | "*text*">;

no arguments
cancels all existing footnotes.

n
specifies the relative line to be occupied by the footnote.

'text' | "text"
specifies the text of the footnote that is enclosed in single or double quotation marks. For compatibility with previous releases, SAS accepts some text without quotation marks. When writing new programs or updating existing programs, *always* surround text with quotation marks.

Details Under OpenVMS, the maximum footnote length is 132 characters. If the footnote length is greater than the value of the LINESIZE= system option, the footnote is truncated to the line size.

See Also

- FOOTNOTE statement in *SAS Language Reference: Dictionary*
- System option: "LINESIZE=" on page 418

%INCLUDE

Includes SAS statements and data lines

Language element: statement
Valid: anywhere in a SAS program
OpenVMS specifics: valid values for *source*, if a file specification is used

Syntax

%INCLUDE *source-1 < . . . source-n>*
</option-list>;

source-1 < . . . source-n>
describes the location of the information that you want to access with the %INCLUDE statement. The three possible sources are an external file specification, previously entered SAS statements from your SAS session, or keyboard entry. The file specification can be any of the file specification forms discussed in "Identifying External Files to the SAS System" on page 166.

This section discusses only external file specifications. For information about including lines from your terminal or from your SAS session, see "Recalling SAS Statements" on page 29 and the SAS statements portion of *SAS Language Reference: Dictionary*.

option-list
specifies portable options for the %INCLUDE statement. For more information about these options, see the %INCLUDE statement in *SAS Language Reference:*

Dictionary. The %INCLUDE statement has no options that are host-specific for the OpenVMS environment.

Details When you execute a program that contains the %INCLUDE statement, the SAS System executes your code, including any statements or data lines that you bring into the program with %INCLUDE.

The %INCLUDE statement is most often used when running SAS in interactive line mode, noninteractive mode, or batch mode. Although you can use the %INCLUDE statement when running SAS using windows, it may be more practical to use the INCLUDE and RECALL commands to access data lines and program statements, and submit these lines again.

The %INCLUDE statement executes statements immediately.

Example

The following is an example of a %INCLUDE statement. Suppose you have issued the following FILENAME statement:

```
filename mypgm '[mydir]program1.sas';
```

Then, in a SAS program you can issue the following %INCLUDE statement to copy in and execute the SAS statements stored in the file PROGRAM1.SAS:

```
%include mypgm;
```

See Also

- %INCLUDE statement in *SAS Language Reference: Dictionary*
- Command: "INCLUDE" on page 230
- RECALL command in SAS online Help
- "Saving SAS Statements" on page 28
- "Recalling SAS Statements" on page 29

INFILE

Specifies an external file to read with an INPUT statement

Language element: statement

Valid: in a DATA step

OpenVMS specifics: valid values for *file-specification*; valid values for *host-option-list*

Syntax

INFILE *file-specification* <*option-list*>
<*host-option-list*>;

file-specification
identifies the source of input data records (usually an external file). It can be any of the file specification forms discussed in "Identifying External Files to the SAS

System" on page 166. The reserved fileref DATALINES allows the INFILE statement to read instream data.

option-list
names portable options for the INFILE statement. For information about these options, see the INFILE statement in *SAS Language Reference: Dictionary*.

host-option-list
names external I/O statement options for the INFILE statement that are specific to the OpenVMS environment. These options can be any of the following:

FAC=

KEY=

KEYVALUE=

LINESIZE=

LRECL=

MBC=

MBF=

RECFM=

SHR=

For an explanation of these options, see "Host-Specific External I/O Statement Options" on page 361 in the FILENAME statement.

You can intersperse options from *option-list* and *host-option-list* in any order.

Note: When using the PIPE device with the INFILE statement, only LRECL is supported. △

Details Because the INFILE statement identifies the file to read, it must execute before the INPUT statement that reads the input data records. You can use the INFILE statement in conditional processing, such as an IF-THEN statement, because it is executable. This allows you to control the source of the input data records.

When you use more than one INFILE statement for the same file specification and you use options in each INFILE statement, the effect is additive. To avoid confusion, use all the options in the first INFILE statement for a given external file.

Example

The following is an example of an INFILE statement:

```
infile food;
```

This INFILE statement uses the default filename form of the file specification (FOOD has not been assigned as a SAS fileref or as an OpenVMS logical name). Therefore, the SAS System looks for the file FOOD.DAT in the current directory.

When the SAS System reads a file, it uses the most recent version by default. For example, if your default directory contains the files FOOD.DAT;1, FOOD.DAT;2, and FOOD.DAT;3, this INFILE statement reads FOOD.DAT;3.

See Also

- INFILE statement in *SAS Language Reference: Dictionary*
- Chapter 7, “Using External Files and Devices,” on page 165
- Statement: “FILENAME” on page 357

LENGTH

Specifies how many bytes the SAS System uses to store a variable’s values

Language element: statement

Valid: in a DATA step

OpenVMS specifics: valid numeric variable lengths

Syntax

LENGTH <*variable-specification-1*>
<. . .*variable-specification-n*><DEFAULT=*n*>;

length
can range from 3 to 8 bytes for numeric variables in the OpenVMS Alpha environment.

The value of *length* can range from 2 to 8 bytes for numeric variables in the OpenVMS VAX environment.

DEFAULT=*n*
changes the default number of bytes used for storing the values of newly created numeric variables from 8 to the value of *n*.

In the OpenVMS Alpha environment, *n* can range from 3 to 8 bytes.

In the OpenVMS VAX environment, *n* can range from 2 to 8 bytes.

Details The LENGTH statement specifies the number of bytes used for storing variables.

In general, the length of a variable depends on

- whether the variable is numeric or character
- how the variable was created
- whether a LENGTH or ATTRIB statement is present.

Subject to the rules for assigning lengths, lengths that are assigned with the LENGTH statement can be changed in the ATTRIB statement and vice versa.

See Also

- LENGTH statement in *SAS Language Reference: Dictionary*
- "Numeric Variables in the Alpha Environment" on page 199
- "Numeric Variables in the VAX Environment" on page 200
- Statement: "ATTRIB" on page 354

LIBNAME

Associates a libref with a SAS data library and lists file attributes for a SAS data library

Language element: statement

Valid: anywhere in a SAS program

OpenVMS specifics: valid values for *engine-name*; specifications for *SAS-data-library*; valid values for *engine/host-option-list*

Syntax

LIBNAME *libref* <*engine*> '*SAS-data-library*'
<*portable-options*> <*engine/host-options*>;

LIBNAME *libref* | _ALL_CLEAR;

LIBNAME *libref* | _ALL_ LIST;

Note: This is a simplified version of the LIBNAME statement syntax. For the complete syntax and its explanation, see the LIBNAME statement in *SAS Language Reference: Dictionary*. △

For an explanation of the LIBNAME statement arguments in the OpenVMS operating environment, see "Associating Librefs" on page 378.

Details The LIBNAME statement associates a libref with a permanent SAS data library and lists the file attributes of a SAS data library.

Note: The LIBNAME statement is also used to clear a libref. For complete documentation on this use, see the LIBNAME statement in *SAS Language Reference: Dictionary*. △

Associating Librefs Use the following form of the LIBNAME statement to associate a libref, an engine, or engine/host options with a SAS data library:

```
LIBNAME libref <engine> 'SAS-data-library'
   <portable-options> <engine/host-options>;
```

libref
is a SAS name that complies with SAS naming conventions and is used in SAS statements to point to *SAS-data-library*. This argument is required.

The *libref* can also be an OpenVMS logical name or a search-string logical name. For more information, see "Using an OpenVMS Logical Name in the

LIBNAME Statement" on page 127 and "Using a Search-String Logical Name to Concatenate SAS Data Libraries" on page 128.

Under OpenVMS, the only reserved librefs are those that are reserved by the SAS System on all operating environments. For a list of reserved librefs, see the LIBNAME statement in *SAS Language Reference: Dictionary*.

engine
tells SAS which engine to use for accessing the library. For a list of valid engine names for OpenVMS, see "Engines Available under OpenVMS" on page 140. The engine that is associated with a libref accesses only files that were created by that engine, not other SAS files.

If you do not specify an engine, then SAS uses the procedures described in "How SAS Assigns an Engine When No Engine Is Specified" on page 130 to assign an engine for you.

SAS-data-library
is the name of the directory that contains the SAS data library, enclosed in quotes. You can omit this argument if you are merely specifying the engine for a libref or an OpenVMS logical name that you previously assigned.

If the directory that you specify does not already exist, then you must create it before you attempt to use the libref that you have assigned to it. (Under OpenVMS, the LIBNAME statement does not actually create directories.)

Use the following syntax for concatenated libraries:

```
LIBNAME libref  ('SAS-data-library' '...'
```

Note that librefs can be used as part of a physical name or a previously assigned libref.

The level of specification depends on your current location in the OpenVMS file structure. For example, if you want to access a directory that is located on another node in your OpenVMS network, then the file specification in the LIBNAME statement must include the node, the device, and the directory levels.

The file specification generally must not extend beyond the directory or subdirectory level (that is, it must not include a filename), because the libref/directory association that is made in the LIBNAME statement gives you access to all SAS files in the data library, not to a single file. However, this rule does not apply if you are assigning a libref for use with the XPORT, OSIRIS, or SPSS engines.

SAS-data-library can also be an OpenVMS logical name (or a path that contains a logical name). In this case, you would be assigning a libref to the logical name, and you would subsequently use the libref in your SAS program. For examples, see "Using an OpenVMS Logical Name in the LIBNAME Statement" on page 127.

Note: Directory wildcard specifications are not supported in LIBNAME statements. If you use an asterisk (*) or an ellipsis (...) in the *SAS-data-library* argument, an error message tells you that the physical name of the library is invalid. △

portable-options
are LIBNAME statement options that are available in all operating environments. For information about these options, see the LIBNAME statement in *SAS Language Reference: Dictionary*.

engine/host-options
are one or more of the following host-specific options:

ALQ=
specifies how many disk blocks to allocate to a new SAS data set. For more information, see the data set option "ALQ=" on page 249.

ALQMULT=
specifies the number of pages that are preallocated to a file. For more information, see the data set option “ALQMULT=” on page 250.

BKS=
specifies the bucket size for a new data set. For more information, see the data set option “BKS=” on page 251.

CACHENUM=
specifies the number of I/O data caches used per SAS file. For more information, see the data set option “CACHENUM=” on page 253.

CACHESIZ=
controls the size of the I/O data cache that is allocated for a file. For more information, see the data set option “CACHESIZ=” on page 253.

DEQ=
tells OpenVMS how many disk blocks to add when it automatically extends a SAS data set during a 'write' operation. For more information, see the data set option “DEQ=” on page 256.

DEQMULT=
specifies the number of pages to extend a SAS file. For more information, see the data set option “DEQMULT=” on page 257.

MBF=
specifies the multibuffer count for a data set. For more information, see the data set option “MBF=” on page 260.

For a complete listing of the available external I/O statement options, see “Summary of External I/O Statement Options” on page 370 in the FILENAME statement.

Not every option is available with every engine. For information about which engine or host options are available with each engine, see Chapter 6, “Using SAS Engines,” on page 139.

All of these options correspond to a data set option of the same name and have the same effect as the data set option. However, engine or host options apply to all SAS data sets that are stored in the SAS data library.

Specify as many options as you need. Separate them with a blank space.

Listing Data Library Attributes You can use the LIBNAME statement to list attributes of SAS data libraries by using the LIST option.

See Also

- LIBNAME statement in *SAS Language Reference: Dictionary*
- “Using the LIBNAME Statement” on page 125
- “Using an OpenVMS Logical Name in the LIBNAME Statement” on page 127
- Statement: “FILE” on page 355
- Statement: “FILENAME” on page 357

SYSTASK

Executes, lists, or kills asynchronous tasks

Valid: anywhere in a SAS program

OpenVMS specifics: all

Syntax

SYSTASK COMMAND "*host-command*"
<XWAIT|NOXWAIT>
<TASKNAME=*taskname*>
<MNAME=*name-var*>
<STATUS=*stat-var*>

SYSTASK LIST <_ALL_ | *taskname*> <STATE> <STATVAR>;

SYSTASK KILL *taskname* <*taskname...*>;

COMMAND
executes the *host-command*.

LIST
lists either a specific active task or all of the active tasks in the system.

KILL
forces the termination of the specified task(s).

host-command
specifies the name of an OpenVMS command (including any command-specific options).

XWAIT | NOXWAIT
determines whether SYSTASK COMMAND suspends execution of the current SAS session until the task has completed. NOXWAIT is the default. For tasks that are started with the NOXWAIT option, you can use the WAITFOR statement when necessary to suspend execution of the SAS session until the task has finished.

TASKNAME=*taskname*
specifies a name that identifies the task. Task names must be unique among all active tasks. A task is *active* if it is running, or if it has completed and has not been waited for using the WAITFOR statement. Duplicate task names generate an error in the SAS log. If you do not specify a task name, SYSTASK will automatically generate a name. If the task name contains a blank character, enclose the task name in quotes.

MNAME=*name-var*
specifies a macro variable in which you want SYSTASK to store the task name that it automatically generated for the task. If you specify both the TASKNAME option and the MNAME option, SYSTASK copies the name that you specified with TASKNAME into the variable that you specified with MNAME.

STATUS=*stat-var*
specifies a macro variable in which you want SYSTASK to store the status of the task. Status variable names must be unique among all active tasks.

ALL
specifies all active tasks in the system.

STATE
displays the status of the task, which can be Start, Failed, Running, or Complete.

STATVAR
displays the status variable associated with the task. The status variable is the variable that you assigned with the STATUS option in the SYSTASK COMMAND statement.

Details

SYSTASK allows you to execute host-specific commands from within your SAS session or application. Unlike the X statement, SYSTASK runs these commands as *asynchronous* tasks, which means that these tasks execute independently of all other tasks that are currently running. Asynchronous tasks run in the background, so you can perform additional tasks while the asynchronous task is still running.

The output from the command is displayed in the SAS log.

Note: Program steps that follow the SYSTASK statements in SAS applications usually depend on the successful execution of the SYSTASK statements. Therefore, syntax errors in some SYSTASK statements will cause your SAS application to abort. △

There are two types of tasks that can be run with SYSTASK:

Task
All tasks started with SYSTASK COMMAND are of type Task. For these tasks, if you do not specify STATVAR or STATE, then SYSTASK LIST displays the task name, type, and state, and the name of the status macro variable. You can use SYSTASK KILL to kill only tasks of type Task.

SAS/Connect Process
Tasks started from SAS/Connect with the SYSTASK BEGIN statement are of type SAS/Connect Process. For SAS/Connect processes, SYSTASK LIST displays the task name, type, and state. You cannot use SYSTASK KILL to kill a SAS/Connect process. For information on starting SAS/Connect processes with SYSTASK, refer to *SAS/CONNECT User's Guide*.

The SYSRC macro variable contains the return code for the SYSTASK statement. The status variable that you specify with the STATUS option contains the return code of the process started with SYSTASK COMMAND. To ensure that a task executes successfully, you should monitor both the status of the SYSTASK statement and the status of the process that is started by the SYSTASK statement.

If a SYSTASK statement cannot execute successfully, the SYSRC macro variable will contain a non-zero value. For example, there could be insufficient resources to complete a task, or the SYSTASK statement could contain syntax errors. With the SYSTASK KILL statement, if one or more of the processes cannot be killed, SYSRC is set to a non-zero value.

When a task is started, its status variable is set to NULL. You can use the status variables for each task to determine which tasks failed to complete. Any task whose status variable is NULL did not complete execution.

Unlike the X statement, you cannot use the SYSTASK statement to start a new interactive session.

TITLE

Specifies title lines for SAS output

Language element: statement

Valid: anywhere in a SAS program

OpenVMS specifics: maximum length of title

Syntax

TITLE <*n*> <'*text*' | "*text*">;

no arguments
cancels all existing titles.

n
specifies the relative line that contains the title line.

'text' | "text"
specifies the text of the title that is enclosed in single or double quotation marks. For compatibility with previous releases, SAS accepts some text without quotation marks. When writing new programs or updating existing programs, *always* surround text with quotation marks.

Details Under OpenVMS, the maximum title length is 132 characters. If the title length is greater than the value of the LINESIZE= system option, then the title is truncated to the line size.

See Also

- TITLE statement in *SAS Language Reference: Dictionary*
- System option: "LINESIZE=" on page 418

X

Issues an operating environment command from within a SAS session

Language element: statement

Valid: anywhere in a SAS program

OpenVMS specifics: operating environment command; OpenVMS subprocesses

Syntax

X <'*DCL-command*'>;

no argument
spawns an OpenVMS subprocess, where you can issue DCL commands.

'DCL-command'
specifies a single DCL command. The value for *DCL-command* must be enclosed in quotation marks.

Details The X statement issues a DCL command from within a SAS session. The SAS System executes the X statement immediately.

For complete information about the X statement, see “Issuing DCL Commands during a SAS Session” on page 36.

WAITFOR

Suspends execution of the current SAS session until the specified tasks finish executing

Valid: anywhere in a SAS program

OpenVMS specifics: all

Syntax

WAITFOR <_ANY | _ALL_> *taskname* <*taskname...*> <TIMEOUT=*seconds*>;

taskname
specifies the name of the task(s) that you want to wait for. The task name(s) that you specify must match exactly the task names assigned through the SYSTASK COMMAND statement. You cannot use wildcards to specify task names.

ANY | _ALL_
suspends execution of the current SAS session until either one or all of the specified tasks finishes executing. The default setting is _ANY_, which means that as soon as one of the specified task(s) completes executing, the WAITFOR statement will finish executing.

TIMEOUT=*seconds*
specifies the maximum number of seconds that WAITFOR should suspend the current SAS session. If you do not specify the TIMEOUT option, WAITFOR will suspend execution of the SAS session indefinitely.

Details

The WAITFOR statement suspends execution of the current SAS session until the specified task(s) finish executing or until the TIMEOUT interval (if specified) has elapsed. If the specified task was started with the XWAIT option, then the WAITFOR statement ignores that task.

For example, the following statements start three different SAS jobs and suspend the execution of the current SAS session until those three jobs have finished executing:

```
systask command "sas myprog1.sas" taskname=sas1;
systask command "sas myprog2.sas" taskname=sas2;
systask command "sas myprog3.sas" taskname=sas3;
waitfor _all_ sas1 sas2 sas3;
```

The SYSRC macro variable contains the return code for the WAITFOR statement. If a WAITFOR statement cannot execute successfully, the SYSRC macro variable will contain a non-zero value. For example, the WAITFOR statement may contain syntax errors. If the number of seconds specified with the TIMEOUT option elapses, then the WAITFOR statement finishes executing, and SYSRC is set to a non-zero value if

- you specify a single task that does not finish executing
- you specify more than one task and the _ANY_ option (which is the default setting), but none of the tasks finishes executing
- you specify more than one task and the _ALL_ option and any one of the tasks does not finish executing.

Any task whose status variable is still NULL after the WAITFOR statement has executed did not complete execution.

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

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

1–58025–526–4

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, by any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

1st printing, October 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

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.