CHAPTER 6

Allocating External Files

Introduction

An external file is any file that is not maintained or structured by SAS and that you use during your SAS session. In a DATA step, you use INFILE and INPUT statements to read data from external files, and you use FILE and PUT statements to write to external files. Some SAS procedures, such as the CPORT and FSLIST procedures, operate directly on individual external files. You can also include external files of SAS program statements in your SAS session by using the INCLUDE command or the %INCLUDE statement.

As with SAS files in CMS, external files are stored within a Shared File System (SFS) file space or on minidisks. See *VM/ESA CMS User's Guide* for information about SFS.

Identifying an External File

To identify the external file to SAS, use a *file-specification* argument in a SAS statement. The *file-specification* argument takes one of the following forms:

external-file
specifies the CMS fileid of the external file and has the following form:

’*filename filetype* <*filemode* | *SFS-directory* | *>’

This syntax can also be used to specify a MACLIB as an aggregate external file, as follows:

’*filename* MACLIB <*filemode* | *’>

To specify a CMS minidisk or SFS directory as an aggregate file, use the following:

’*filemode* | *SFS-directory* | *’

Omitting the filename and filetype specifies an aggregate file, which consists of an SFS directory or minidisk. Files in the directory or minidisk are accessed as independent members of the aggregate file.

If you omit *filemode* or *SFS-directory*, or if you specify an asterisk (*) when you reference the file for input, accessed disks are searched in the standard CMS search order, and the first occurrence of a file with a matching filename and filetype is read. If you omit *filemode* or *SFS-directory*, or if you specify an asterisk (*) when you reference the file for output, the file is written on the first R/W disk.

You cannot use an asterisk (*) for either *filename* or *filetype*.

fileref
specifies a logical name that is associated with an external file. This name contains 1 to 8 characters and is not enclosed in quotes. The first character must be a letter (A to Z or a to z) or an underscore (_); the remaining characters can be any combination of letter, numbers, or underscores. If a SAS FILENAME statement, FILENAME function, or CMS FILEDEF command has assigned this logical name to an external file, then the device or file that is identified by that FILENAME statement is used. If a FILENAME statement has not been issued, then SAS creates an assignment for the fileref by creating a CMS file specification, and SAS uses the fileref as the filename and defaults for filetype and filemode.

fileref(member)
specifies an independent member of an aggregate file that was previously associated with the fileref.

Aggregate External Files

An aggregate external file is processed as a single file that contains multiple independent members. For input only, CMS MACLIBs can be specified as aggregate files. For input and update, an aggregate file can be an SFS directory or a CMS minidisk.

The FILENAME statement is used to specify an aggregate external file:

FILENAME *fileref* ’*filename filetype filemode* | *SFS-directory* | *’;

Concatenating External Files

To concatenate external files or directories, use the FILENAME statement. To concatenate files that contain multiple SAS programs in order to invoke the programs in a single invocation, use the SYSIN= system option.

Aggregate Files Compared with Concatenated Files

Aggregate files contain independent files that are processed as individual members. Concatenated files appear to SAS as a single file that contains the data of all of the files that are listed in concatenation. In other words, concatenated files are not independent of each other. The order of the concatenation means that files that are listed first in the FILENAME concatenation take precedence over files that appear later in the list.

Aggregate files can be concatenated. SAS treats an aggregate concatenation as it does any other, by searching each aggregate in turn, in order of appearance, in the initial concatenation specification.

Accessing an External File

To access an external file, specify its filename in a SAS statement or command. For example, this INCLUDE command accesses a file that contains SAS statements and includes it into the Program Editor window:

```
include 'mycode sas a'
```

These statements access an external file of data that is used to create a SAS data set:

```
data mydata;
     infile 'rawdata data b';
```

If you plan to use the same external file several times in your SAS program, it is more efficient to use the FILENAME statement to establish a fileref for the file. (See "Advantages of Using the FILENAME Statement" on page 62.) You can subsequently use the fileref to refer to the file instead of specifying the filename again.

Note: Use the CMS FILEDEF command to assign a DDname, which is also a logical name, only when you read OS/390 sequential or partitioned data sets on OS/390 disks that are accessed by shared DASD, or when you read OS/390 simulated CMS files that are identified by filemode number 4. △

Using the FILENAME Statement to Reference External Files

A fileref that is established by a FILENAME statement or FILENAME function remains in effect until the SAS session ends, or until it is changed or deleted by a FILENAME statement that specifies the same fileref. A FILENAME statement for disk always overrides a CMS FILEDEF command for disk. When you use a FILENAME statement to assign a fileref to a disk file, the native CMS interface is used for I/O. If you use a CMS FILEDEF command to assign a fileref to a disk file, OS/390 Simulation Services (provided by CMS) are used.

CAUTION:

Do not assign different filerefs to the same physical file or use the same fileref for concurrent access. For example, if you assign a fileref to a file, then browse the file through the FSLIST window, do not attempt to go to the Program Editor window and submit a DATA step to write to the file while it is still displayed in the FSLIST window. △

Advantages of Using the FILENAME Statement

There are several advantages to using the FILENAME statement to identify external files.

- It is portable. The FILENAME statement is recognized by SAS under all operating environments. You can develop a SAS program under CMS and port it to another operating environment with fewer changes to your program statements than if you used the CMS FILEDEF command.
- It is easier for novice CMS users to understand than the CMS FILEDEF command.
- It enables you to list the filerefs that you have assigned.
- It reduces processing time.
  - Filerefs that are assigned by a FILENAME statement are found before filerefs that are assigned that are by a FILEDEF command.
  - Filerefs that are assigned by the FILENAME statement use OS/390 Simulation only for nondisk files. Filerefs that are assigned by a FILEDEF command always use OS/390 Simulation, even for disk files.

FILENAME Statement Syntax

This section provides a brief overview of FILENAME statement syntax. For complete information about the FILENAME statement, see "FILENAME" on page 227. The general form of the FILENAME statement is

FILENAME *fileref* | _ALL_ *device-type* <'*external-file*'> <*options*>;

The FILENAME statement takes the following options:

fileref
: is a logical name by which the external file is referenced. The fileref must begin with a letter or underscore and must contain 1 to 8 characters consisting of letters, numbers, or underscores.

ALL
: is a reserved fileref that is used only to list or clear filerefs.

device-type
: specifies the type of output or input device for the file. If device-type not specified, SAS assumes that the file is on disk.

'*external-file*'
: identifies the physical file to be associated with the fileref.

options
: is a list of options that control how the file is read or written. Use a blank space to separate each option when you specify more than one. All options use a *keyword=value* format.

FILENAME Statement Examples

- The following FILENAME statements associate the fileref MYDATA with the external file HOUSES DATA A on disk. Note that the second example is equivalent to the first, but the disk option is unnecessary.

  ```
  filename mydata 'houses data a';


  filename mydata disk 'houses data a';
  ```

- The following FILENAME statement associates the fileref OUTDATA with the external file SOME FILE, in the FPOOL:USER3.DATA97 SFS directory.

  ```
  filename outdata
     'some file fpool:user3.data97';
  ```

- The following FILENAME statement associates the fileref INDATA with the external file SURVEY MARCH on disk. If this fileref is used for output, it is written to the first R/W disk. If this fileref is used for input, the standard CMS minidisk search order is followed.

  ```
  filename indata 'survey march';
  ```

- The following FILENAME statement associates the fileref OUT with the virtual disk PUNCH. Any output to OUT is directed to the virtual disk PUNCH.

  ```
  filename out punch;
  ```

- The following FILENAME statement associates the fileref MYLIB with the tape device defined as 182.

  ```
  filename mylib tape 'tap2' ;
  ```

- The following FILENAME statement associates the fileref MYDIR with the SFS directory FPOOL:MYUSER.MYDIR as an aggregate external file:

  ```
  filename mydir 'fpool:myuser.mydir';
  ```

- The following FILENAME statement clears the association between the fileref MYDATA and any device or file, provided that the fileref was originally assigned by a FILENAME statement. CLEAR is assumed by default. However, you may state it explicitly.

  ```
  filename mydata clear;
  ```

- The following FILENAME statement concatenates three files. When the fileref corresponding to a list of files to be concatenated occurs in a later open-for-read operation, the data is read sequentially. When the fileref occurs in a later open-for-write operation, output is written to the first file in the list.

  ```
  filename all
     ('mar file' 'apr file' 'may file');
  ```

- The following FILENAME statement specifies multiple members in multiple MACLIBs. You cannot specify that output be sent to a MACLIB file.

  ```
  filename a
     ('one maclib' 'two maclib' 'three maclib');
  ```

Using CMS Pipelines

You can use a standard CMS pipeline specification in place of the external file specification in the FILE, FILENAME, and INFILE statements, and in the FILENAME function. This enables you to receive input from any CP or CMS command or pipeline device driver, or to route output to any pipeline device driver.

For the FILENAME statement and FILENAME function, specify PIPE as the device type. For the FILE and INFILE statements, specify PIPE as an option. If you specify PIPE as an option in a FILE or INFILE statement, you cannot specify any other options in that statement.

SAS supports inline comments as an extension of the standard CMS pipeline interface, as shown:

```
filename adata pipe
   '(name adata)    /* name pipeline */
   < archive data a /* read in file  */
   | unpack         /* unpack        */
   ';
```

Data transfer between SAS and the CMS pipeline takes place in the pipeline's FITTING stage. By default, SAS inserts a FITTING stage at the beginning of pipeline specifications that are used for output and at the end of pipeline specifications that are used for input. For example, when this ADATA fileref is referenced in an INFILE statement, the pipeline specification is internally coded as

```
'(name adata) < archive data a | unpack | fitting sasio'
```

You can override the default insertion of the FITTING stage by explicitly inserting the stage into your pipeline specification. This is allowed as long as the FITTING stage is always either the first or last stage of a stream. For example, the FANIN stage in the middle of this pipeline delivers data to the FITTING stage:

```
filename twostrm pipe
   '(endchar %) < jan data a
   | a: fanin
   |    fitting sasio
   %
    < feb data a
   | a:
   ';
```

You may prefer to write complex pipelines as REXX pipeline stages, then to specify the REXX stage name in your SAS program. For the unpack example above, you could write a file called UNPKDATA REXX:

```
/* UNPKDATA REXX */
parse arg fileid
'callpipe <' fileid,
    '| unpack',
    '| *:'
exit
```

Then you could write your FILENAME statement as

```
filename mydata pipe
   '(name mydata)    /* name pipeline */
   rexx unpkdata archive data a
     /* read and unpack a file        */
   ';
```

Macro variables are resolved within a pipeline specification that is enclosed in double quotes.

This example illustrates the use of pipelines with INFILE, FILENAME, and FILE statements, and with macro variables.

```
/* Data step to get spoolid. Only accept files        */
/* sent by a specified userid                          */
data _NULL_;
   infile                    /* define the pipeline   */
      'cp query rdr          /* query all rdr files   */
         |strfind /USERABC / /* from this userid only */
         |take 1             /* only process one file */
         |specs word 2 1'    /* keep only the spoolid */
      PIPE;
   input spid $;     /* read spoolid as char variable */
                    /* save spoolid as macro variable */
   call symput('SPOOLID',spid);
run;

/* define fileref for reading netdata rdr file        */
filename nd pipe
   "reader file &spoolid  /* read the file            */
        |drop 1           /* skip TAG                 */
        |specs 2-* 1      /* skip CC byte             */
        |deblock netdata  /* interpret netdata format */
        |strfind xC0      /* keep only data records   */
        |specs 2-* 1      /* skip control byte        */
        ";

/* data step to read input from rdr file              */
/* write data to USERABC DATA in packed format        */
data abc;
   file                   /* pipe for PUT processing  */
      '  strip            /* strip blanks             */
         | pack V 80      /* write packed format      */
         | > userabc data a fixed' pipe; /* this file */
   infile nd;             /* read from ND fileref     */
   input x;               /* read var x from each rec */
   put x;   /* write to USERABC DATA in packed format */
run;
```

Listing Current Filerefs

In a windowing environment SAS session, you can issue the FILENAME command from the command line to display the ACTIVE FILE SHORTCUTS window. This window lists all current filerefs that are assigned by SAS FILENAME statements and the complete CMS fileid to which each fileref is assigned.

You can use the following form of the FILENAME statement in any SAS job to obtain the same information that is presented in the FILENAME window:

```
filename _all_ list;
```

Specifying the Physical Attributes of a File

You can specify values for file characteristics in the *options* argument in the FILE, FILENAME, and INFILE statements. Note that you do not need to specify file characteristics when you are concatenating files, since the existing characteristics of the files are used.

The following section explains which values are used if you specify file characteristics for the same file in multiple places.

Order of Precedence for Input

If you use a fileref to reference a file for input, any values that you specify for file characteristics in the INFILE statement take precedence over any that you have specified in a FILENAME statement or in a CMS FILEDEF command. Any values that you specify for file characteristics in a FILENAME statement or CMS FILEDEF command take precedence over the file characteristics of the existing file. If you do not specify any values for file characteristics, then the values for the existing file are used.

Order of Precedence for Output

If you use a fileref to reference a file for output, any values that you specify for file characteristics in the FILE statement take precedence over any that you have specified in a FILENAME statement or in a CMS FILEDEF command. If you do not specify any values for file characteristics when you are appending to an existing file, the values for the existing file are used. If you do not specify any values for file characteristics when you are writing to a new file, the defaults are used. Default file characteristics are shown in “FILENAME” on page 227.

Options

The following file characteristics can be specified in the *options* argument in a FILE, FILENAME, or INFILE statement. All options use a *keyword=value* format, and multiple options are separated by blank spaces. For the FILE statement, note that options cannot be specified if you use FILE LOG, FILE PRINT, or FILE PIPE. For the FILENAME statement, note that if file characteristics are given in a statement that specifies several files to be concatenated, then all files must have the same attributes.

BLKSIZE=*value*

specifies the buffer size that is allocated to contain records. Valid values are 1 through 65535.

Records do not have to be blocked. However, because disk or tape activity is reduced when records are blocked, it is more efficient to use blocks. Blocking records is different when you use CMS Native I/O Services than it is when you use OS/390 Simulation Services.

For CMS Native I/O Services, the following apply:

- No blocking of records is performed in variable-length files. The BLKSIZE= option is ignored, and RECFM=VB is treated the same as RECFM=V.
- For fixed-length files, BLKSIZE= specifies the maximum buffer size to contain records. The default block size is the value of LRECL= times the number of logical-record-length records that fit into 1,024 bytes. For example,

a fixed-length file with LRECL=80 by default has BLKSIZE=960. Twelve 80-byte records can fit into 1,024 bytes, taking up 960 bytes.

- CMS does not support blocking for a *device-type* of TERMINAL.

For OS/390 Simulated I/O Services, the following apply:

- Blocking is not supported for a *device-type* of READER.
- Blocking sizes that are specified for *device-type* of TAPE, as well as OS/390 shared DASD files, must match the block size of the existing file.
- When RECFM=FB, the value of the LRECL= option is the length of the longest record, and BLKSIZE is an integer multiple of LRECLs. For example, if the longest record is 70 bytes, then LRECL=70 and BLKSIZE=70 x *n*, where *n* is an integer.
- When RECFM=VB, the value of the LRECL= option is the length of the longest record plus 4 bytes that hold record descriptor information. The value of the BLKSIZE= option can be any value up to 32,760, as long as it is at least 4 bytes longer than the value of the LRECL= option. (These 4 bytes are the block descriptor information.)
- Any time the value of the RECFM= option includes A (which specifies a PRINT file), add 1 to the value of the LRECL= option in case the column contains carriage-control characters.

DENSITY=*value*
specifies tape density in bits per inch. Valid values include 200, 556, 800, 1600, 6250, and 38K.

DISP=*value*
specifies the status (disposition) of the file. Acceptable values are

MOD	specifies that output lines are to be written after any existing lines in the file.
OLD	specifies that any output lines are to be written at the beginning of the file. This is the default. For CMS, this option is identical to the NEW option.
NEW	specifies that any output lines are to be written at the beginning of the file. For CMS, this option is identical to the OLD option.

FILEVAR=*variable*
enables you to dynamically change input and output files in the middle of a DATA step.

LABEL=*value*
indicates the type of label processing for a tape file. LABOFF is the default. Valid values are

- BLP
- LABOFF
- NL
- SL

If SL, NL, or BLP is specified, then an additional label value *n* can be specified after the SL, NL, or BLP. The value of *n* indicates the file position in a multifile volume. The default value is 1.

LEAVE=YES
indicates that a multifile tape is not repositioned at open for LABOFF or BLP processing. For SL tapes, LEAVE=YES does not reposition before label processing.

Omitting LEAVE or specifying LEAVE=NO causes a tape to be rewound and repositioned each time a file is opened. (See “Working with SAS Files on Tape” on page 36 for details on tape processing.)

LRECL=*value*
specifies the logical record length in bytes. Valid values are 1 through 65,535.

RECFM=*format*
specifies the format of records in the file.
You can choose one of the following record formats:

F	specifies fixed-length records, unblocked.
FB	specifies fixed-length records, blocked.
V	specifies variable-length records, unblocked.
VB	specifies variable-length records, blocked.
U	specifies undefined record format. For CMS disk files, this is the same as V.

You can use the following values in any of the previously listed formats except U.

A	specifies that the first byte of each record is an ANSI printer control character and that the file is to be handled as a print file.
S	specifies that the file contains spanned records (V), or that the file contains standard blocks (F). FS, FBS, VS, and VBS files must be assigned with a CMS FILEDEF command.

SYSPARM=*value*
passes an option string to a tape management system for standard label tapes. If you type a question mark (?) SAS prompts you for option settings.

TRACK=*value*
specifies the tape setting. Valid values are

- 7TRACK
- 9TRACK
- 18TRACK
- 3489B
- 3490C
- 3590B
- 3590C
- XF

VOLID=*value*
specifies the volume serial number to be verified in the tape. If the value contains any special characters, it must be enclosed in single quotes.

See the FILEDEF command entry in *VM/ESA CMS Command Reference* for more information about these options.

SAS does not interfere with FILEDEF options that are already specified. Therefore, TAPE options that are specified in the FILE, FILENAME, or INFILE statements that conflict with an existing FILEDEF to TAPE are ignored. See “Working with SAS Files on Tape” on page 36 for more information.

If a variable-length file is opened for update (possibly in the DATA step) and if the replacement line is not the same length as the existing line, then the standard CMS file system action truncates the file at that point. No message is given. The next READ to that file returns an EOF message.

Sending E-Mail from Within SAS

SAS lets you send electronic mail (e-mail) by using the DATA step, procedures, or SCL. Sending e-mail from within SAS allows you to

- □ use the logic of the DATA step or SCL to subset e-mail distribution based on a large data set of e-mail addresses
- □ send e-mail automatically upon completion of a SAS program that you submitted for batch processing
- □ direct output through e-mail based on the results of processing.

SAS e-mail is implemented in the following language elements:

- □ The EMAILSYS= system option, which specifies the name of the external CMS pipeline stage that delivers mail messages to the host mail system. See “Using the Mail System Interface and the EMAILSYS= System Option” on page 69 for details.
- □ The FILE and FILENAME statements, which are used to specify the name of an e-mail fileref and the mailing instructions, or directives, that are used to send it. See “FILENAME Statement Syntax for Electronic Mail” on page 70 for details.
- □ The PUT statement, which is used in the DATA step or SCL to create the e-mail message and to specify or change mailing directives. See “PUT Statement Syntax for E-Mail” on page 71 for details.

To send an e-mail message from a SAS session, use these e-mail language elements in a DATA step, in a procedure, or in SCL code (see “Using the DATA Step, Procedures, or SCL Code to Send E-Mail” on page 73).

Using the Mail System Interface and the EMAILSYS= System Option

SAS sends all e-mail through an external CMS pipeline stage that is written as a REXX exec. The name of the exec is SASMAIL REXX, and it is stored on the SAS system disk.

Note: The SASMAIL REXX exec may need to be customized with site-specific information that is pertinent to your mail system. Comments at the beginning of the exec guide you through the customizations. Work with your system administrator and local SAS Support Consultant as needed. △

You can use your own CMS pipeline stage instead of SASMAIL by specifying the name of your exec as the value of the EMAILSYS= system option. You can specify a value for EMAILSYS= in your SAS configuration file, in your SAS invocation command, or during your SAS session by using the OPTIONS statement or OPTIONS window. The syntax of the OPTIONS statement is as follows:

EMAILSYS=’*stage-specification*’

The stage specification can be of the form *filename filetype filemode*. If the filetype is omitted, SAS assumes a filetype of REXX. If the filemode is omitted, SAS assumes a filemode of ‘*’. Quotes are required around the entire specification if anything more than a filename is specified.

FILENAME Statement Syntax for Electronic Mail

To send electronic mail from within a SAS session, issue a FILENAME statement of the following form:

FILENAME *fileref* EMAIL '*address*' <*e-mail-options*>;

The arguments are defined as follows:

fileref
is a valid fileref.

'*address*'
is the e-mail address of the user to whom you want to send e-mail. This argument is optional, but you must specify at least one address, either here or with the TO= option or with the !EM_TO! PUT directive.

e-mail-options
can be any of the following:

TO=*to-address*
specifies the primary recipients of the e-mail message. If an address contains special characters or more than one word, enclose the address in single or double quotes, as follows:

```
to='joe@hisplace.org'
```

To specify more than one address, enclose the group of addresses in parentheses and each address in single or double quotes, as follows:

```
to=('joe@hisplace.org' 'jane@herplace.org')
```

To specify the recipient's name along with the address, enclose the address in angle brackets, as follows:

```
to="Joseph Smith <joe@hisplace.org>"
```

A recipient can also be specified as a nickname to be resolved from your NAMES file.

Specifying the TO= option overrides the *address* argument.

CC=*cc-address*
specifies the copy recipients of the e-mail message. If an address contains special characters or more than one word, enclose it in single or double quotes, as follows:

```
cc='joe@hisplace.org'
```

To specify more than one address, enclose the group of addresses in parentheses and each address in single or double quotes, as follows:

```
cc=('joe@hisplace.org' 'jane@herplace.org')
```

To specify the recipient's name along with the address, enclose the address in angle brackets, as follows:

```
cc="Joseph Smith <joe@hisplace.org>"
```

A recipient can also be specified as a nickname to be resolved from your NAMES file.

SUBJECT='*subject*'
specifies the subject of the message. If the subject text contains special characters or is longer than one word, enclose it in single or double quotes, as follows:

```
subject=Sales
```

```
subject='June Report'
```

Any subject not enclosed in quotes is converted to uppercase.

ATTACH='*file-specification*'
: specifies, inside single or double quotes, one or more files to attach to the message. Specify filename and filetype and either the filemode or SFS directory name, as follows:

```
attach='opinion txt'
```

The filemode is '*' by default. To attach more than one file, enclose the group of file specifications in parentheses, as follows:

```
attach=('june98 txt .reports.june' 'july98 txt .reports.july')
```

TYPE='*content-type*'
: specifies the content-type of the message, as follows:

```
type='text/plain'
type='text/html'
type='image/gif'
```

The TYPE value must be enclosed in quotes.

EMAILSYS='*stage-specification*'
: specifies the name of a CMS pipeline stage to be used to send e-mail. This FILENAME option overrides the value of the EMAILSYS= system option, which is SASMAIL REXX by default.

EMAILID='*from-user*'
: specifies the e-mail address that will appear on the From: header of the e-mail message. The default value is the userid and host name of the user running SAS. You may prefer to specify a different address for a server application. If the e-mail address contains special characters or more than one word, enclose the entire specification in quotes. To specify a name along with the address, enclose the address in angle brackets, as follows:

```
emailid='Joseph Smith <joe@hisplace.org>'
```

You can also specify the *e-mail-options* in the FILE statement inside the DATA step. Options that you specify in the FILE statement override any corresponding options that you specified in the FILENAME statement.

PUT Statement Syntax for E-Mail

In your DATA step, after using the FILE statement to define your e-mail fileref as the output destination, use PUT statements to define the body of the message.

You can also use PUT statements to specify e-mail directives that change the attributes of your electronic message or perform actions with it. Specify only one directive in each PUT statement; each PUT statement can contain only the text that is associated with the directive that it specifies.

The directives that change the attributes of your message are as follows:

!EM_TO! *addresses*
: replaces the current primary recipient addresses with *addresses*. For example:

```
PUT "!EM_TO!" "joe@hisplace.org";
```

or:

```
user="joe@hisplace.org";
put '!EM_TO!' user;
```

To specify more than one address, enclose the list of addresses in parentheses and enclose each address in single or double quotes, as follows:

```
PUT "EM_TO!" '("joe@hisplace.org" "jane@herplace.org")';
```

or:

```
list='("joe@hisplace.org" "jane@herplace.org")';
put "!EM_TO!" list;
```

If you want to specify a recipient's name along with the e-mail address, include the address in angle brackets, as follows:

```
user='Joseph Smith <joe@hisplace.org>'
PUT "EM_TO!" user;
```

A recipient can also be specified as a nickname that will be resolved from your NAMES file.

!EM_CC! *addresses*
replace the current copied recipient addresses with *addresses*. For example:

```
PUT "!EM_CC!" "joe@hisplace.org";
```

or:

```
user="joe@hisplace.org";
put '!EM_CC!' user;
```

To specify more than one current copied recipient, enclose a list of addresses in parentheses and enclose each address in single or double quotes, as follows:

```
PUT "!EM_CC!" '("joe@hisplace.org" "jane@herplace.org")';
```

or:

```
list='("joe@hisplace.org" "jane@herplace.org")';
put '!EM_CC!' list;
```

If you want to specify a recipient's name along with the address, enclose the address in angle brackets, as follows:

```
ccuser='Joseph Smith <joe@hisplace.org>';
PUT "!EM_CC!" ccuser;
```

A recipient can also be specified as a nickname that will be resolved from your NAMES file.

!EM_SUBJECT! *subject*
replaces the current subject of the message with *subject*.

!EM_CONTENTTYPE! *content-type*
replaces the current content-type of the message with *content-type*.

!EM_ATTACH! *file-specifications*
replaces the names of any attached files with *file-specifications*. For each attachment, specify filename and filetype and either the filemode or the SFS directory name. The default filemode is '*', as follows:

```
PUT '!EM_ATTACH!' 'opinion txt';
```

To attach more than one file, enclose each file specification in single or double quotes and enclose the list of file specifications in parentheses, as follows:

```
mycfg='sasv8 config *';
syscfg='sasv8sys config *';
PUT '!EM_ATTACH!' '("'mycfg'"'  '"'syscffg'")';
```

The directives that perform actions are as follows:

!EM_SEND!
: sends the message with the current attributes. By default, the message is automatically sent at the end of the DATA step. If you use this directive, SAS sends the message when it encounters the directive, and sends it again at the end of the DATA step.

!EM_ABORT!
: aborts the current message. You can use this directive to stop SAS from automatically sending the message at the end of the DATA step.

!EM_NEWMSG!
: clears all attributes of the current message, including TO, CC, SUBJECT, TYPE, ATTACH, and the message body.

Using the DATA Step, Procedures, or SCL Code to Send E-Mail

In general, a DATA step, procedure, or SCL code that sends e-mail has the following components:

- a FILENAME statement that specifies the EMAIL device-type
- FILENAME or FILE statements that contain e-mail options that indicate the e-mail recipients, subject, and any attached files
- for a DATA step or in SCL code, PUT statements that create the body of the message
- for a DATA step or in SCL code, PUT statements that contain special e-mail directives that can override the e-mail attributes (TO, CC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and start a NEWMSG)
- for procedures, ODS statements or graphics options that direct output to the EMAIL fileref.

Sending E-Mail from the DATA Step

Suppose that you want to share a copy of your SASV8 CONFIG file with your coworker Jim, whose user ID is JBrown. You could send it by submitting the following DATA step:

```
filename mymail email 'JBrown'
         subject='My SASV8 CONFIG file'
         attach='sasv8 config';

data _null_;
   file mymail;
   put 'Jim,';
   put 'This is my SASV8 CONFIG file.';
   put 'I think you might like the
       new options I added.';
```

```
run;
```

The following example sends a message and two attached files to multiple recipients. It specifies the e-mail options in the FILE statement instead of in the FILENAME statement:

```
filename outbox email 'ron@acme.com';

data _null_;
   file outbox
      to=('ron@acme.com' 'lisa@acme.com')
      /* Overrides value in */
      /* filename statement */

      cc=('margaret@yourcomp.com'
         'steve@abc.com')
      subject='My SAS output'
      attach=('results listing' 'code sas')
      ;
   put 'Folks,';
   put 'Attached is my output from the
      SAS program I ran last night.';
   put 'It worked great!';
run;
```

You can use conditional logic in the DATA step to send multiple messages and to control which recipients get which message. For example, suppose that you want to send customized reports to members of two different departments. Your DATA step might look like the following:

```
filename reports email 'Jim';

data _null_;
   file reports;
   infile cards eof=lastobs;
   length name dept $ 21;
   input name dept;

   /* Assign the TO attribute           */
   put '!EM_TO!' name;

   /* Assign the SUBJECT attribute      */
   put '!EM_SUBJECT! Report for ' dept;
   put name ',';
   put 'Here is the latest report for ' dept '.';

   /* ATTACH the appropriate report     */
   if dept='marketing' then
      put '!EM_ATTACH! mktrept txt';
   else
      put '!EM_ATTACH! devrept txt';

   /* Send the message */
   put '!EM_SEND!';

   /* Clear the message attributes       */
```

```
      put '!EM_NEWMSG!';
      return;

    /* Abort the message before the RUN     */
    /* statement causes it to be sent again */
   lastobs: put '!EM_ABORT!';

     datalines;
   Susan          marketing
   Jim            marketing
   Rita           development
   Herb           development
   ;
   run;
```

The resulting e-mail message and its attachments are dependent on the department to which the recipient belongs.

Note: You must use the !EM_NEWMSG! directive to clear the message attributes between recipients. The !EM_ABORT! directive prevents the message from being automatically sent at the end of the DATA step. △

Sending Procedure Output as E-Mail

Procedures that write to filerefs can be used to send e-mail. The following example shows how to use the Output Delivery System (ODS) to send HTML output in e-mail:

```
filename outbox email
   to=susan
   type='text/html'
   subject='Temperature conversions'
   ;

data temperatures
   do centigrade = --40 to 100 by 10;
      fahrenheit = centigrade*9/5+32;
      output;
   end;
run;

ods html
   body=outbox /* Mail it! */
   rs=none;

title 'Centigrade to Fahrenheit Conversion Table';
proc print;
id centigrade;
var fahrenheit;
run;

ods html close;
```

The following example shows how to create and send a GIF image in e-mail.

```
filename gsasfile email
   to=Jim
   type='image/gif'
```

```
      subject="SAS/GRAPH output"
      ;

   goptions dev=gif gsfname=gsasfile;

   proc gtestit pic=1; run;
```

Sending E-Mail Using SCL Code

The following example is the SCL code behind a frame entry design for e-mail. The frame entry includes the following text entry fields that let the user enter information:

mailto	the user ID to send mail to
copyto	the user ID to copy (CC) the mail to
attach	the name of a file to attach
subject	the subject of the mail
line1	the text of the message

The frame entry also contains a pushbutton called SEND that causes this SCL code (marked by the **send:** label) to execute.

```
   send:

      /* set up a fileref */

      rc = filename('mailit','userid','email');

      /* if the fileref was successfully set up
         open the file to write to */

      if rc = 0 then do;
         fid = fopen('mailit','o');
         if fid > 0 then do;

            /* fput statements are used to
               write the mail and the components,
               such as subject, address, etc. */

            fputrc1  = fput(fid,line1);
            rc = fwrite(fid);

            fputrc2  = fput(fid,'!EM_TO! '||mailto);
            rc = fwrite(fid);
            fputrc3  = fput(fid,'!EM_CC! '||copyto);
            rc = fwrite(fid);

            fputrc4  = fput(fid,'!EM_ATTACH! '||attach);
            rc = fwrite(fid);
            fputrc5  = fput(fid,'!EM_SUBJECT! '||subject);
            rc = fwrite(fid);

            closerc  = fclose(fid);
         end;
      end;
```

```
return;

cancel:
   call execcmd('end');
return;
```

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

SAS® Companion for the CMS Environment, Version 8

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

ISBN 1–58025–481–0

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

1st printing, October 1999