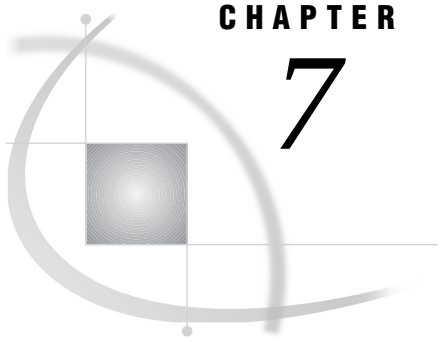


CHAPTER 7

Using Groupware to Distribute SAS Data

Introduction to Using Groupware with the SAS System

The term *groupware* refers to software applications that you use to communicate and share data with colleagues within your organization. The SAS System under OS/2 enables you to use groupware to share SAS data. In particular, you can share data by using both electronic mail (e-mail) and Lotus Notes (a client-server groupware and messaging application). See your groupware and Lotus documentation for complete details about this software.

Sending E-Mail from within the SAS System

The SAS System lets you send e-mail either interactively (using a dialog box) or programmatically (using SAS statements in a DATA step or SCL). SAS supports the Vendor Independent Mail (VIM—such as Lotus cc:Mail and Lotus Notes) interface.

To send e-mail from within SAS, select the **File** pull-down menu and then select **Send...**. Your e-mail software provides the interface to send mail.

Note: You need an e-mail program that supports VIM before you can take advantage of SAS e-mail support. Also, although you can use the SAS System to send messages, you must use your e-mail program to view or read messages. △

Initializing E-Mail

The SAS e-mail feature supports the following SAS system options in the SAS configuration file or in your SAS session:

-EMAILSYS VIM
specifies the Vendor Independent Mail interface.

Note: The directory that contains the e-mail DLL file (for example, VIM32.DLL) must be specified in your OS/2 PATH environment variable. The SAS System uses the first e-mail DLL that it finds for the interface that you specify. Your installed e-mail support must use 32–bit architecture. △

-EMAILDLG SAS
specifies to use the e-mail interface that is provided by the SAS System.

-EMAILID "*name*"
specifies your e-mail login. If *name* contains spaces, you must enclose it in double quotes.

-EMAILPW "*password*"
specifies your e-mail login password, where *password* is the login password for your login name. If *password* contains spaces, you must enclose it in double quotes.

For example, if your login ID is **J.B. Smith** and your password is **rosebud**, your SAS invocation might look like this:

```
c:\sas\sas.exe -emailsys vim
               -emailid "J.B. Smith"
               -emailpw rosebud
```

Note: If you don't specify the EMAILID and EMAILPW system options at invocation (and you are not otherwise signed on to your e-mail system already), the SAS System will prompt you for them when you initiate e-mail. △

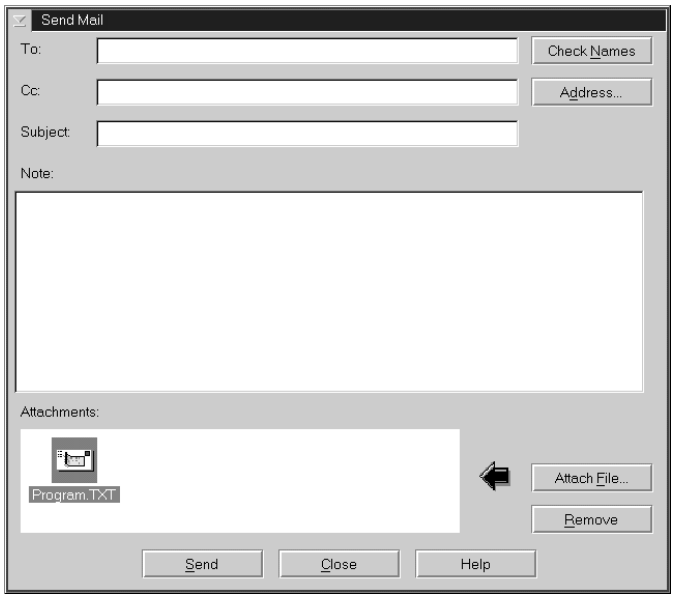
You must have a 32–bit e-mail client program to initiate e-mail from the SAS System.

Using the Send Mail Dialog Box to Send E-Mail

The default value of the EMAILDLG system option is SAS. The default lets you send e-mail by using a dialog box that is provided by the SAS System.

You can access the Send Mail dialog box by selecting the **Send...** item from the **File** menu. Display 7.1 on page 131 shows the Send Mail dialog box.

Display 7.1 Send Mail Dialog Box



The following are descriptions of the fields in the Send Mail dialog box:

To:
the primary recipients of your e-mail. You must specify one or more e-mail addresses that are valid for your mail system before you can send e-mail. Separate multiple recipients with a semicolon (;).

Cc:
the e-mail addresses of those who will receive a copy of the mail that you are sending. You can leave this field blank if you want. Separate multiple recipients with a semicolon (;).

Subject:
the subject of your message. You can leave this field blank if you want.

Note:
the body of your message. You can copy text from SAS application windows or other OS/2 applications and paste it here (using the CTRL+C and CTRL+V accelerator key combinations). If your note text exceeds the window space provided, you can scroll backward and forward by using the arrow keys or the PgUp and PgDn keys.

Some e-mail systems limit the note length to 32K (or 32,768 characters). Text that you enter in the **Note:** area will automatically wrap at the right side.

For large amounts of text, attach a text file as described below.

Attachments:
icons and names of any files that you want to send with the message. You or the recipient can open an attached file by double-clicking on its icon, provided that its file extension has a File Manager association with an OS/2 application (for example, the .TXT extension might be associated with a text editor).

Use the Attach File... button to open a file selection dialog box that you can use to select files to attach. To remove an attachment, select the file's icon in the **Attachments** field and click on Remove.

Note: The attached files are sent as they exist on the disk; that is, if you edit a file before attaching it to an e-mail message, the *saved* version of the file is sent with the message. △

To verify that the addresses that you specified in the **To:** and **Cc:** fields are valid, click on Check Names. If one or more of the addresses is ambiguous (that is, the mail program cannot locate them in the address books) the SAS System displays an error message and highlights the first ambiguous address.

Note that an ambiguous address is not necessarily invalid. It is possible to send mail to recipients outside your immediate local-area network (LAN) using gateways, whose addresses might not be resolved by using Check Names.

Whether an address is considered invalid or ambiguous depends on the e-mail program that you are using and on the configuration of your network. For example, suppose that you want to send e-mail to a colleague on the Internet. Your LAN might have a gateway to the Internet so that you can address the mail to **JBrown@rhythm.com at Internet** (where **at** is the gateway directive keyword and **Internet** is the name of a gateway on your LAN). Because your e-mail program uses the **at** keyword to direct your message to the **Internet** gateway, the address is considered valid. However, when you click on Check Names, the address is considered ambiguous because the final destination address cannot be resolved by using the local address book. You can still click on Send to send the message without an error.

Clicking on Address... invokes the address book facility for your e-mail program, provided that the facility is accessible.

Using the DATA Step or SCL to Send E-Mail

Using the EMAIL access method, you can use the DATA step or SCL to send e-mail from within the SAS System. This has several advantages:

- You can use the logic of the DATA step or SCL to subset e-mail distribution based on a large data set of e-mail addresses.
- You can automatically send e-mail upon completion of a SAS program that you submitted for batch processing.
- You can direct output through e-mail based on the results of processing.
- You can send e-mail messages from within a SAS/AF FRAME application, by customizing the user interface.

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

- a FILENAME statement with the EMAIL device-type keyword
- options that are specified in the FILENAME or FILE statement and that indicate the e-mail recipients, subject, and any attached files
- PUT statements that contain the body of the message
- PUT statements that contain special e-mail directives (of the form !EM_*directive*!) that can override the e-mail attributes (TO, CC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and NEWMSG).

To send e-mail by using the DATA step or SCL, you must be signed on to your e-mail program. For more information about how to use your e-mail program with the SAS System, see “Sending E-Mail from within the SAS System” on page 129.

Syntax

FILENAME *fileref* EMAIL '*address*' <*email-options*>;

where

fileref
is a valid fileref.

EMAIL
is the device-type keyword that indicates that you want to use e-mail.

'*address*'
is the valid destination e-mail address of the user that you want to send mail to. You can specify 'nul' here, and then specify the destination addresses in *email-options*.

email-options
can be any of the following:

Note: Each e-mail option can be specified only in a FILENAME statement that overrides the corresponding SAS system option. △

EMAILID="*name*"
specifies your e-mail login ID or the profile that is used to access the underlying e-mail system. If *name* contains a space, enclose it in double quotes. You can specify this e-mail option in the FILENAME statement in order to override the SAS system option.

EMAILPW="*password*"
specifies your e-mail login password, where *password* is the login password for your login name. If *password* contains a space, enclose it in double quotes.You can specify this e-mail option in the FILENAME statement in order to override the SAS system option.

EMAILSYS=VIM
specifies the e-mail interface:

VIM
Vendor Independent Mail, such as Lotus Notes or cc:Mail, which is the default.

TO=*to-address*
specifies the primary recipients of the e-mail. If an address contains more than one word, you must enclose the address in double quotes. If you want to specify more than one address, you must enclose the group of addresses in parentheses and each address in double quotes. For example, **to="John Smith"** and **to=("J. Callahan" "P. Sledge")** are valid TO values.

CC=*cc-address*
specifies the recipients that you want to receive a copy of the e-mail. If an address contains more than one word, you must enclose the address in double quotes. If you want to specify more than one address, you must enclose the group of addresses in parentheses and each address in double quotes. For example, **cc="John Smith"** and **cc=("J. Callahan" "P.Sledge")** are valid CC values.

SUBJECT=*subject*
specifies the subject of the message. If the subject text is longer than one word (that is, it contains at least one blank space), you must enclose the text in double quotes. For example, **subject=Sales** and **subject="June Report"** are valid subjects.

ATTACH=*filename.ext*
specifies the full path name of one or more files to attach to the message. If you want to attach more than one file, you must enclose the group of filenames in parentheses and each filename in double quotes. For example,

attach=opinion.txt and **attach=("june94.txt" "july94.txt")** are valid file attachments.

You can also specify *email-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.

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

You can also use PUT statements to specify e-mail directives that change the attributes of your message or that perform actions with it. You can 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. Here are the directives that change your message attributes:

!EM_TO! *addresses*
replaces the current primary recipient addresses with *addresses*. If a single address contains more than one word, you must enclose that address in quotes. If you want to specify more than one address, you must enclose each address in quotes and the group of addresses in parentheses.

!EM_CC! *addresses*
replaces the current copied recipient addresses with *addresses*. If you want to specify more than one address, you must enclose each address in quotes and the group of addresses in parentheses.

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

!EM_ATTACH! *filename.ext*
replaces the names of any attached files with *filename.ext*. If you want to specify more than one file, you must enclose each filename in quotes and the group of filenames in parentheses.

Here are the directives that perform actions:

!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, the SAS System sends the message when it encounters the directive, and again at the end of the DATA step. This directive is useful for writing DATA step programs that conditionally send messages or use a loop to send multiple messages.

!EM_ABORT!
aborts the current message. You can use this directive to stop the SAS System from automatically sending the message at the end of the DATA step. By default, SAS sends a message for each FILE statement.

!EM_NEWMSG!
clears all attributes of the current message that were set by using PUT statement directives.

Example: Sending E-Mail from the DATA Step

Suppose you want to share a copy of your SAS configuration file with your coworker Jim, whose user ID is **JBrown**. Example Code 7.1 on page 134 shows how to send the file with the DATA step.

Example Code 7.1 Sending a File with the Data Step

```
filename mymail email "JBrown"
         subject="My SASV8.CFG file"
```

```
         attach="c:\sas\sasV8.cfg";
data _null_;
 file mymail;
 put 'Jim,';
 put 'This is my SAS configuration file.';
 put 'I think you might like the';
 put 'new options I added.';
run;
```

Example Code 7.2 on page 135 sends a message and attaches a file to multiple recipients, and specifies the e-mail options in the FILE statement instead of the FILENAME statement:

Example Code 7.2 Attaching a File and Specifying Options in the FILE Statement

```
filename outbox email "Ron B";
data _null_;
 file outbox
       /* Overrides value in */
       /* filename statement */
    to=("Ron B" "Lisa D")
    cc=("Margaret Z" "Lenny P")
    subject="My SAS output"
    attach="c:\sas\results.out"
    ;
 put 'Folks,';
 put 'Attached is my output from the SAS';
 put 'program I ran last night.';
 put 'It worked great!';
run;
```

You can use conditional logic in the DATA step to send multiple messages and control which recipients get which message. For example, suppose that you want to send customized reports to members of two different departments. Example Code 7.3 on page 135 shows such a DATA step.

Example Code 7.3 Sending Customized Messages Using the Data Step

```
filename reports email "Jim";
data _null_;
 file reports;
 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 '.'
 if dept='marketing' then
   put '!EM_ATTACH! c:\mktrept.txt';
 else  /* ATTACH the appropriate report */
   put '!EM_ATTACH! c:\devrept.txt';

     /* Send the message. */
 put '!EM_SEND!';
    /* Clear the message attributes.*/
```

```
 put '!EM_NEWMSG!';
    /* Abort the message before the */
    /* RUN statement causes it to   */
    /* be sent again.               */
 put '!EM_ABORT!';
 cards;
 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. △

Example Code 7.4 on page 136 shows how to send a message and to attach a file to multiple recipients, and it specifies the e-mail options in the FILENAME statement instead of the FILE statement. Specifying these options overrides the SAS system options EMAILID=, EMAILPW=, and EMAILSYS=.

Example Code 7.4 Overriding SAS System Options

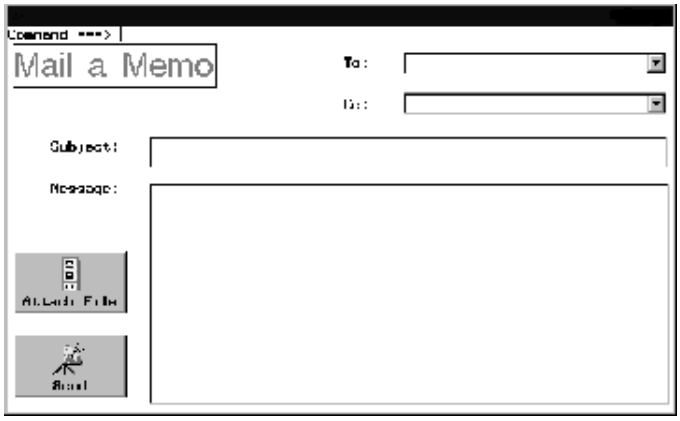
```
filename outbox email "Ron B" emailsys=VIM
 emailpw="mypassword" emailid="myuserid";
data _null_;
 file outbox
       /* Overrides value in */
       /* filename statement */
    to=("Ron B" "Lisa D")
    cc=("Margaret Z" "Lenny P")
    subject="My SAS output"
    attach="c:\sas\results.out"
    ;
 put 'Folks,';
 put 'Attached is my output from the SAS';
 put 'program I ran last night.';
 put 'It worked great!';
run;
```

.

Example: Sending E-Mail Using SCL Code

The following example is the SCL code that is behind a FRAME entry that is designed for e-mail. The FRAME entry might look similar to the one shown in Display 7.2 on page 137.

Display 7.2 An Example E-Mail FRAME Entry



The FRAME entry has objects that let the user enter the following 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.

Example Code 7.5 on page 137 shows the FRAME entry. This entry contains a pushbutton called SEND that invokes this SCL code (marked by the **`send:`** label).

Example Code 7.5 Invoking SCL Code from a FRAME Entry

```
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
          implement writing the mail and
          the components such as subject,
          who to mail to, 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;
```

Populating a Lotus Notes Database Using the DATA Step and SCL Code

Client and Server Implementations

SAS provides the access engine NOTESDB to enable client users to add new Notes documents to an existing Notes database. This engine is compatible with Lotus Notes Release 4.x and later (32-bit only).

Version 8 supports Notes Server Version 4.x or higher (32-bit only). However, because the NOTESDB access engine communicates directly with the client version of Notes, there are no restrictions on the types of platforms on which the Notes Server can run.

Creating New Notes Documents

Using the NOTESDB access engine and the DATA step or SCL code, you can create new Notes documents. However, the engine cannot create a new database. To create a Notes document, make sure that a client version of Lotus Notes and a valid user ID certification are installed on the machine that will be using the NOTESDB engine.

Note: Be sure that the Notes directory is in the system path. △

Lotus Notes does not have to be running in order for the SAS System to access it. You will be prompted for a password to access the Notes server through the SAS System.

In general, DATA step or SCL code that interacts with a Notes database has the following components:

- a FILENAME statement with the NOTESDB device-type keyword
- PUT statements that contain data directives and the data to place in the Notes database
- PUT statements that contain action directives to control when to send the data to the Notes database.

Syntax for Populating a Lotus Notes Database

FILENAME *fileref* NOTESDB;

where

fileref
is a valid fileref.

NOTESDB
is the device-type keyword that indicates that you want to use a Lotus Notes database.

In your DATA step, use PUT statements with data directives to define which database you want to use and the data that you want to send.

Note: Although the directives that you specify to access a Notes database are not case-sensitive, the fields that you specify using those directives are. Also, only one

directive per PUT statement is permitted. Each directive should be delimited with an exclamation point and surrounded with single quotes. △

The data directives you can use are

!NSF_SERVER! *server-name*
indicates the Notes server to access, where *server-name* represents a Lotus Notes server. If you do not specify this directive, SAS uses your local system as the source for the databases. If you specify this directive more than once, the server that is specified in the most recent PUT statement is used.

Note: If you attempt to access a Notes server through SAS, you will be prompted for your password to the server. △

!NSF_DB! *database-filename*
indicates the Notes database file to access. When you access a database locally, SAS looks for the database in the Notes data directory. If it is not found there, SAS searches the system path. Alternatively, you can specify the fully qualified path for the database. You must specify a Notes database file with this directive before you can access a Notes database from SAS. If you specify this directive more than once, the database that is specified in the most recent PUT statement is used.

!NSF_FORM! *form-name*
specifies the form that Notes should use when displaying the added note. If this directive is not specified, Notes uses the default database form. If you specify this directive more than once, the form that is specified in the most recent PUT statement with the !NSF_FORM! directive is used.

!NSF_ATTACH! *filename*
attaches a file to the added note. SAS looks for the file in the Notes data folder. If it is not there, SAS searches the system path. Alternatively, you can specify the fully qualified path for the file. You can attach only one file in a single PUT statement with the !NSF_ATTACH! directive. To attach multiple files, use separate PUT statements with !NSF_ATTACH! directives for each file.

!NSF_FIELD! *field-name*!*field value*
adds the value to the field name that is specified. SAS detects the correct format for the field and formats the data accordingly. Note that SAS deletes all line feeds or carriage returns; you should not insert any of these control characters since they affect the proper display of the document in Notes. Multiple PUT statements with the !NSF_FIELD! directive and the same field name concatenate the information in that field. Also, PUT statements with no directives are concatenated to the last field name that was submitted, or they are ignored if no PUT statements with !NSF_FIELD! directives have previously been submitted.

You can populate fields, which can be edited, of the following types:

- text
- numeric
- keywords
- bitmap (in OS/2 bitmap format) by using the following form:

!NSF_FIELD! *field-name* <*bitmap-filename*>

Use these directives to perform actions on the Notes database:

!NSF_ADD!
adds a document to the Notes database within the DATA step program.

!NSF_ABORT!
indicates not to add the note when SAS closes the data stream. By default, the driver attempts to add a note at the end of a SAS program for every FILE statement that is used. This directive negates this behavior.

!NSF_CLR_FIELDS!
clears all the field values that were held and that use the !NSF_FIELD! directive. This directive in conjunction with !NSF_ADD! facilitates writing DATA step programs with loops that add multiple notes to multiple databases.

!NSF_CLR_ATTACHES!
clears all the field values that were held and that use the !NSF_ATTACH! directive. This directive in conjunction with !NSF_ADD! facilitates writing DATA step programs with loops that add multiple notes to multiple databases.

Note: The contents of PUT statements that do not contain directives are concatenated to the data associated with the most recent field value. △

Examples of Populating Lotus Notes Databases

Example Code 7.6 on page 140 uses the Business Card Request database that is supplied by Lotus Notes. This DATA step creates a new document in the database and supplies values for all of its fields.

Example Code 7.6 Using the Business Card Request Database

```
01 filename reqcard NOTESDB;
02 data _null_;
03  file reqcard;
04  put '!NSF_DB! examples\buscard.nsf';
05  put '!NSF_FIELD!Status! Order';
06  put '!NSF_FIELD!Quantity! 500';
07  put '!NSF_FIELD!RequestedBy! Systems';
08  put '!NSF_FIELD!RequestedBy_CN! Jane Doe';
09  put '!NSF_FIELD!NameLine_1! Jane Doe';
10  put '!NSF_FIELD!NameLine_2! Developer';
11  put '!NSF_FIELD!AddressLine_1! Software R Us';
12  put '!NSF_FIELD!AddressLine_2! 123 Silicon Lane';
13  put '!NSF_FIELD!AddressLine_3! Garner, NC 27123';
14  put '!NSF_FIELD!AddressLine_4! USA';
15  put '!NSF_FIELD!PhoneLine_1! (910) 777-3232';
16 run;
```

Line 1 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 3 uses the assigned fileref to direct output from the PUT statement. Line 4 indicates which Notes database to open. Lines 5 to 15 specify the field and the value for that field for the new Notes document that is being created. Status is the field name and Order is the value that is placed in the Status field for the particular document. Line 16 executes these SAS statements. A new Notes document is created in the Business Card Request database.

Example Code 7.7 on page 140 uses each observation in the SALES data set to create a new document in the Quarterly Sales database and fills in the **`Sales`**, **`Change`**, and **`Comments`** fields for the documents.

Example Code 7.7 Creating a New Document from a Data Set

```
01 data sasuser.sales;
02    length comment $20;
03    format comment $char20.;
04    input sales change comment $ 12-31;
05 cards;
```

```
06 123472 342 Strong Increase
07 423257 33  Just enough
08 218649 4   Not high enough
09 ;
10 run;
11 filename sales NOTESDB;
12 data _null_;
13    file sales;
14    set sasuser.sales;
15    put '!NSF_DB! qrtsales.nsf';
16    put '!NSF_FORM! Jansales';
17    put '!NSF_ADD!';
18    put '!NSF_FIELD!Sales !' sales;
19    put '!NSF_FIELD!Change!' change;
20    put '!NSF_FIELD!Comments!' comment;
21    put '!NSF_CLR_FIELDS!';
22 run;
```

Line 11 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 13 uses the assigned fileref to direct the output from the PUT statement. In line 15, the NSF_DB data directive indicates which Notes database to open. Lines 18, 19, and 20 specify the field and its value for the new Notes document that is being created. **Sales** is the field name and **sales** is the value that is placed in the **Status** field for the particular document. Line 22 executes these SAS statements. A new Notes document is created in the Business Card Request database.

Expanding on the Business Card Request database example, you can create multiple Notes documents within a single DATA step or SCL code by using action directives as well as data directives. Example Code 7.8 on page 141 shows how to create multiple Notes documents within a single DATA step.

Example Code 7.8 Creating Multiple Notes Documents within a Single DATA Step

```
01 filename reqcard NOTESDB;
02 data _null_;
03   file reqcard;
04   put '!NSF_DB!Examples\buscard.nsf';
05   put '!NSF_FIELD!Status! Order';
06   put '!NSF_FIELD!Quantity! 500';
07   put '!NSF_FIELD!RequestedBy!Systems';
08   put '!NSF_FIELD!RequestedBy_CN! Jane Doe';
09   put '!NSF_FIELD!NameLine_1! Jane Doe';
10   put '!NSF_FIELD!NameLine_2! Developer';
11   put '!NSF_FIELD!AddressLine_1! Software R Us';
12   put '!NSF_FIELD!AddressLine_2! 123 Silicon Lane';
13   put '!NSF_FIELD!AddressLine_3! Garner, NC 27123';
14   put '!NSF_FIELD!AddressLine_4! USA';
15   put '!NSF_FIELD!PhoneLine_1! (910) 555-3232';
16   put '!NSF_ADD!';
17   put '!NSF_CLR_FIELDS!';
18   put '!NSF_FIELD!Status! Order';
19   put '!NSF_FIELD!Quantity! 10';
20   put '!NSF_FIELD!RequestedBy! Research and Development';
21   put '!NSF_FIELD!RequestedBy_CN! John Doe';
22   put '!NSF_FIELD!NameLine_1! John Doe';
23   put '!NSF_FIELD!NameLine_2! Analyst';
```

```
24   put '!NSF_FIELD!AddressLine_1! Games Inc';
25   put '!NSF_FIELD!AddressLine_2! 123 Software Drive';
26   put '!NSF_FIELD!AddressLine_3! Cary, NC 27511';
27   put '!NSF_FIELD!AddressLine_4! USA';
28   put '!NSF_FIELD!PhoneLine_1! (910) 555-3000';
29   run;
```

Line 1 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 3 uses the assigned fileref to direct the output from the PUT statement. Line 4 indicates which Notes database to open. Lines 5 to 15 specify the field and the value for that field for the new Notes document that is being created. **Status** is the field name and **Order** is the value placed in the **Status** field for this particular document. Line 16 forces the creation of a new Notes document. Line 17 clears the values for the fields that are used with the !NSF_FIELD! data directives in the previous lines. Lines 18 to 28 specify the field and the value for that field for the second Notes document that is being created. **Status** is the field name, and **Order** is the value that is placed in the **Status** field for the second document. Line 29 executes these SAS statements. A second Notes document is created in the Business Card Request database.

Only one !NSF_DB! data directive is issued in the preceding example. By default, the second Notes document is created in the same database as that referenced in the !NSF_DB! data directive on line 4. In order to create the second Notes document in another database, you must issue another !NSF_DB! data directive with the new database filename prior to the execution of line 18. The key additions to this example are the action directives on lines 16 and 17.

Note: Not all directives are case-sensitive. However, the values that follow the data directives, such as form name and field name, are case-sensitive. △

Preparing SAS/GRAPH Output for a Notes Document

SAS/GRAPH output can be passed to a Notes document through the NOTESDB access engine. A slight variation of the syntax for the !NSF_FIELD! data directive enables SAS/GRAPH output to be directed to a rich text format field in a Notes document. Follow these steps:

- Export the SAS/GRAPH output to a bitmap file format.
- Use the modified !NSF_FIELD! data directive syntax to assign the value of the bitmap filename to an RTF field. Use the following syntax:

  !NSF_FIELD! *RTF-field-name* < *bitmap-filename*

Example Code 7.9 on page 142 uses the modified syntax.

Note: This example uses the Electronic Library sample database. △

Example Code 7.9 Exporting SAS/GRAPH Output into a Notes Document

```
01  title1 'US Energy Consumption for 1955-1988';
02  proc gplot data=3Dsampsio.energy1;03;
03  plot consumed*year / des=3D'D0319U01-
04        1';04  run;
05  dm 'graph1; export "c:\usenergy.bmp"
    "format=3DBMP"';
06  quit;
07
```

```
08  filename newdoc NOTESDB;
09  data _null_;
10   file newdoc;
11   put '!NSF_DB!Examples\hrdocs.nsf';
     put '!NSF_FIELD!Subject! US Energy Consumption'
13   put '!NSF_FIELD!Categories! Office Services';
14   put '!NSF_FIELD!Body! US Energy Consumption for
          1955-1988';
15   put '!NSF_FIELD!Body<c:\usenergy.bmp';
16  run;
```

Lines 1 to 6 contain code that is taken from the SAS/GRAPH samples by using a sample data set to generate SAS/GRAPH output. Line 8 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 10 uses the assigned fileref to direct the output from the PUT statement. Line 11 indicates which Notes database to open. Lines 12 to 14 specify the field and the value for that field for the new Notes document that is being created. **`Subject`** is the field name and **`US Energy Consumption`** is the value that is placed in the **`Subject`** field for this particular document. Line 15 indicates a display of **`usenergy.bmp`** bitmap file in the **`Body`** field because the **`<`** rather than the exclamation point (**`!`**) is used to separate the field value from the field name. Line 16 executes these SAS statements. A new Notes document is created in the Electronic Library database.

In the preceding example, the **`Detailed`** field is an RTF field. When using RTF fields, you can intersperse data and bitmaps.

Retrieving Information from Preexisting Notes Documents

The SAS/ACCESS to ODBC pass-through engine enables you to retrieve information about existing Notes documents in a Notes database. Example Code 7.10 on page 143 shows how to use the DATA step to retrieve information from the Business Card Request database.

Example Code 7.10 Using ODBC to Retrieve Information from Preexisting Notes Documents

```
01  proc sql;
02     connect to ODBC ("dsn=3Dbuscard");
03     create table sasuser.buscard as
04     select * from connection to
05     ODBC (select * from All_Requests_By_Organization);
06     disconnect from ODBC;
07  run;
```

Line 1 processes SQL statements to manipulate SQL views and tables. Line 2 connects to ODBC, which establishes a connection to Notes through the SAS/ACCESS ODBC driver and the NotesSQL ODBC driver by using the 3Dbuscard data source. Lines 3, 4, and 5 create a table and the permanent SQL table, sasuser.buscard, from the data that are retrieved from the Notes Business Card Request database table. The table is called **`All_Requests_By_Organization`**. This is the default view that is assigned to the Business Card Request database. Line 6 disconnects from ODBC and closes the connection to the Notes database. Line 7 executes these SAS statements. A new data set named **`buscard`** is created in the SASUSER library.

As another alternative, you may view the available tables within Notes databases by using the SQL Query Window. The SQL Query Window, a component of SAS, is an interactive interface that allows you to easily build queries without being familiar with SQL. You can invoke it by issuing the QUERY command from the command line.

For more information about the SQL procedure, see *SAS Procedures Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OS/2® Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999. 448 pp.

SAS® Companion for the OS/2® Environment, Version 8

ISBN 1-58025-521-3

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

1st printing, October 1999