

## CHAPTER

## 7

## Using Groupware to Distribute SAS Data

<i>Introduction to Using Groupware with the SAS System</i>	161
<i>Sending Electronic Mail from within the SAS System</i>	161
<i>Initializing Electronic Mail</i>	162
<i>Using the Send Mail Dialog Box to Send Electronic Mail</i>	162
<i>Using the DATA Step or SCL to Send Electronic Mail</i>	164
Syntax	164
<i>Example: Sending E-Mail from the DATA Step</i>	167
<i>Example: Sending E-Mail Using SCL Code</i>	169
<i>Populating a Lotus Notes Database Using the DATA Step and SCL Code</i>	170
<i>Creating New Notes Documents</i>	170
Syntax for Populating a Lotus Notes Database	170
Examples of Populating Lotus Notes Databases	172
<i>Preparing SAS/GRAPH Output for a Notes Document</i>	174
<i>Using SAS with the NotesSQL ODBC Driver</i>	175
<i>Retrieving Information from Pre-Existing Notes Documents</i>	176

### Introduction to Using Groupware with the SAS System

*Groupware* is a term used to describe the software applications that you use to communicate and share data with colleagues and other interested parties within your organization. The SAS System under Windows facilitates the use of groupware to share SAS data. In particular, SAS enables you share data 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 Electronic Mail from within the SAS System

The SAS System lets you send electronic mail either interactively (using a dialog box) or programmatically (using SAS statements in a DATA step or SCL). SAS supports these types of electronic mail interfaces: MAPI (such as Microsoft Exchange) and Vendor Independent Mail (VIM—such as Lotus cc:Mail and Lotus Notes).

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

*Note:* You need an electronic mail program that supports one of these protocols before you can take advantage of the SAS System's electronic mail support. Also, although you can use the SAS System to send messages, you must use your electronic mail program to view or read messages. △

---

## Initializing Electronic Mail

The electronic mail feature of the SAS System supports the following SAS system options in the SAS configuration file or when invoking your SAS session:

**-EMAILSYS MAPI | VIM**

specifies which e-mail interface to use. By default, the SAS System uses MAPI.

*Note:* The directory that contains the e-mail DLL file (for example, MAPI32.DLL or VIM32.DLL) must be specified in your Windows PATH environment variable. The SAS System will use the first e-mail DLL that it finds for the interface you specify. Your installed e-mail support must use 32-bit architecture.  $\Delta$

**-EMAILDLG NATIVE | SAS**

specifies whether to use the native e-mail interactive dialog box provided by your e-mail application or the e-mail interface provided by the SAS System. You can use the native dialog box with the SAS System only if the e-mail system supports the MAPI interface. SAS uses the native dialog box by default.

**-EMAILID "name"**

specifies your VIM e-mail login ID or MAPI profile used to access the underlying e-mail system. 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:

```
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 logged in to your e-mail system already), the SAS System will prompt you for them when you initiate electronic mail.  $\Delta$

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 Electronic Mail

The default value of the EMAILDLG system option is NATIVE, which lets you take advantage of your own e-mail software when sending e-mail from within SAS. If the value of the EMAILDLG system option is set to SAS, you can send electronic mail using a dialog box provided by the SAS System. If you are using the native e-mail interface provided by your e-mail system vendor, this discussion about the Send Mail dialog box does not apply to you. For more information about using that interface, see the documentation from your e-mail vendor.

You can access the Send Mail dialog box by selecting the **Send...** item from the **File** menu. Display 7.1 on page 163 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 any users that you want to receive a copy of the mail 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.

**Note:**

the body of your message. You can copy text from SAS application windows or other Windows 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 using the arrow keys or the PgUp and PgDn keys.

Some e-mail systems currently 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 a Windows application (for example, the .TXT extension might be associated with Notepad).

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.  $\triangle$

To verify whether the addresses 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 of your immediate 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 you are using and on the configuration of your network. For example, suppose you want to send e-mail to a colleague on the Internet. Your LAN might have a gateway to the Internet such 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 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 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 the facility is accessible.

---

## Using the DATA Step or SCL to Send Electronic Mail

Using the EMAIL access method, you can use the DATA step or SCL to send electronic 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, customizing the user interface.

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

- a FILENAME statement with the EMAIL device-type keyword
- options specified in the FILENAME or FILE statement indicating 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 start a NEWMSG).

To send e-mail using the DATA step or SCL, you must be signed on to your e-mail program. For more information about how to use your electronic mail program with the SAS System, see “Sending Electronic Mail from within the SAS System” on page 161.

### 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 you want to send mail to. You can specify 'nul' here, and then specify the destination addresses in the *email-options*.

*email-options*

can be any of the following:

*Note:* Each e-mail option can be specified only in a FILENAME statement which overrides the corresponding SAS system option.  $\Delta$

EMAILID="*name*"

specifies your e-mail login ID or profile that is used to access the underlying e-mail system. If you specify MAPI in the EMAILSYS e-mail option, specify your profile name. If *name* contains a space, enclose it in double quotes. This e-mail option can be specified in the FILENAME statement which overrides 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. This e-mail option can be specified in the FILENAME statement which overrides the SAS system option.

EMAILSYS=MAPI | VIM

supports two types of electronic mail interfaces:

MAPI

interface supported by Windows and Windows NT, which is used by Microsoft Exchange. MAPI is the default.

VIM

Vendor Independent Mail, such as Lotus or cc:Mail.

TO=*to-address*

specifies the primary recipients of the electronic 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 you want to receive a copy of the electronic 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 the *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.

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. You can specify only one directive in each PUT statement; each PUT statement can contain only the text associated with the directive 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 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 167 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 167 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 you want to send customized reports to members of two different departments. Example Code 7.3 on page 167 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.  $\Delta$

Example Code 7.4 on page 168 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. This has the effect of overriding the values for 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;

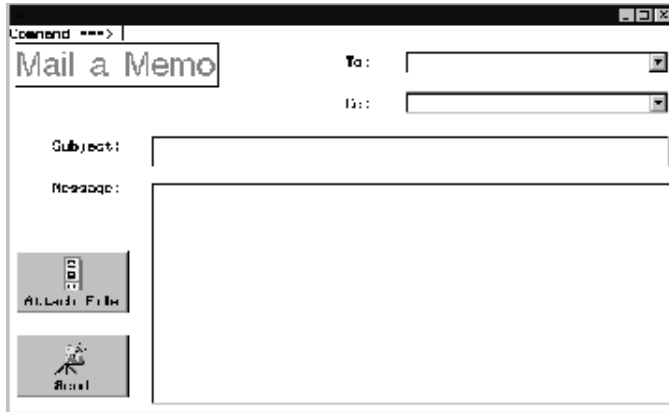
```



## Example: Sending E-Mail Using SCL Code

The following example is the SCL code behind a FRAME entry designed for e-mail. The FRAME entry might look similar to the one shown in Display 7.2 on page 169.

**Display 7.2** An Example E-Mail FRAME Entry



The FRAME entry has objects 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.

Example Code 7.5 on page 169 shows the FRAME entry which also 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. */
      fputc1= fput(fid,line1);
      rc = fwrite(fid);
      fputc2= fput(fid,'!EM_TO! '||mailto);
      rc = fwrite(fid);
      fputc3= fput(fid,'!EM_CC! '||copyto);
      rc = fwrite(fid);
      fputc4= fput(fid,'!EM_ATTACH! '||attach);
      rc = fwrite(fid);

```

```

    fputc5= 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

SAS provides an 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 (32-bit only) for the SAS System Version 8.

---

### Creating New Notes Documents

The NOTESDB access engine enables users to add data to an existing Notes database. However, the NOTESDB access engine does not provide a way for retrieving information from existing Notes documents. The NotesSQL ODBC driver allows for retrieval of data from Notes databases. See “Retrieving Information from Pre-Existing Notes Documents” on page 176 for these details.

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. A client version of Lotus Notes and a valid Notes user ID certification must be installed on the machine that will be using the NOTESDB engine.

*Note:* The Notes directory must be in the system path.  $\Delta$

However, Lotus Notes does not have to be running in order for the SAS System to access it. The user 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 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 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 accessing 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 specified in the most recent PUT statement is used.

**!NSF\_FORM!** *form-name*

specifies the form 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 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 directory. 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 specified. SAS detects the correct format for the field and formats the data accordingly. Note that SAS extracts all line feeds or carriage returns; you should not insert any of these control characters as they affect the proper display of the document in Notes. Multiple PUT statements with the !NSF\_FIELD! directive and the same field name will concatenate the information in that field. Also, PUT statements with no directives are concatenated to the last field name 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

You may add text which will be formatted by Lotus Notes. You may also add a bitmap (in Windows bitmap format) 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 at that point in time within the DATA step program.

**!NSF\_ABORT!**

indicates to abort from adding the note when closing the data stream. By default, the driver attempted to add a note at the end of a SAS program for every FILE statement used. This directive negates this behavior.

**!NSF\_CLR\_FIELDS!**

clears all the field values that were held using the !NSF\_FIELD! directive. This directive in conjunction with !NSF\_ADD! is used to facilitate writing DATA step programs with loops that add multiple notes to multiple databases.

**!NSF\_CLR\_ATTACHES!**

clears all the field values that were held using the !NSF\_ATTACH! directive. This directive in conjunction with !NSF\_ADD! is used to facilitate 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.  $\Delta$

## Examples of Populating Lotus Notes Databases

Example Code 7.6 on page 172 uses the Business Card Request database 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 173 uses each observation in the SALES data set to create a new document in the **qrtsales.nsf** 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 173 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 5-15 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 18-28 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 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 would have to 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:* All directives are not case sensitive. However, the values following the data directives, such as form name and field name, are case sensitive.  $\Delta$

---

## 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. The procedure is:

- 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. Syntax is:

**!NSF\_FIELD! RTF-field-name < bitmap-filename**

Example Code 7.9 on page 174 uses the modified syntax.

*Note:* This example uses the Electronic Library sample database.  $\Delta$

**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';
12     put '!NSF_FIELD!Subject! US Energy Consumption';
13     put '!NSF_FIELD!Categories! Office Services';
14     put '!NSF_FIELD!Body! US Energy Consumption for
15         1955-1988';
16     put '!NSF_FIELD!Body<c:\usenergy.bmp';
16 run;

```

Lines 1-6 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 12-14 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 ! 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.

---

## Using SAS with the NotesSQL ODBC Driver

SAS also provides a SAS/ACCESS to ODBC pass-through engine that allows users to retrieve information about existing Notes documents in a Notes database.

*Note:* This assumes that a Notes ODBC driver and datasource have been configured.  $\Delta$

Both Windows and Windows NT 4.0 with Service Pack 3 require the following software:

*Note:* Installation of Service Pack 3 for Windows NT 4.0 upgrades the ODBC drivers to version 3.0 (3.0.23.1 driver manager). This may cause other ODBC 2.0 drivers to fail. Consult your third-party vendor's ODBC reference.  $\Delta$

- Lotus Notes Client Version 4.x (32-bit only)
- ODBC Driver NotesSQL 2.04 (2.04.0203) (32-bit).

NotesSQL is an ODBC driver provided by Lotus. It can be downloaded free of charge from <http://www.lotus.com>.

After you have the software, you must:

- 1 Set up the NotesSQL ODBC driver.

Lotus provides a file (.nfs) that explains how to set up the driver.

- 2 Configure the ODBC datasource.

You must complete the Lotus Notes ODBC 2.0 Setup screen. Add the appropriate information to these fields:

*Note:* Examples are in parentheses.  $\Delta$

- Data Source name ( **buscard**)
- Description ( **Test Notes Access**)
- Server or Database name ( **c:\notes\data\buscard.nsf**).
- NotesSQL Options Setup, which contains these fields:
  - Max Length of Text Fields (254)
  - Max Number of Tables (20)
  - Max Number of Subqueries (20)
  - Max Length of Rich Text Fields (512)
  - SQL Statement Length (4096)
  - Thread Timeout in seconds (60).

Click  after you have completed the Setup screen.

---

## Retrieving Information from Pre-Existing Notes Documents

The SAS/ACCESS to ODBC pass-through engine allows users to retrieve information about existing Notes documents in a Notes database. Example Code 7.10 on page 176 shows an example of 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 Pre-existing 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 to ODBC driver and the NotesSQL ODBC driver by using the 3Dbuscard data source. Lines 3, 4, and 5 create a table and sasuser.buscard from the data that is retrieved from the Notes Business Card Request database table which 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 PROC SQL, see *SAS Procedures Guide*.



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp.555.

**SAS Companion for the Microsoft Windows Environment, Version 8**

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

ISBN 1-58025-524-8

All rights reserved. Printed in the United States of America.

**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).

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

1st printing, September 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.