An Everyday Guide to Versions 7 and 8 of the SAS® System

Susan J. Slaughter, Independent Consultant, Davis, CA
Lora D. Delwiche, IT/ANSA, University of California, Davis, CA

What is an everyday guide?

The new versions of the SAS System introduce many important innovations. But with so many advanced and specialized features, it's easy to miss the fact that there are also many features to help beginners and occasional users too.

This paper describes the changes and additions that we believe every SAS user should know—features that will change the way you use SAS every day. So, whether you are a beginner learning SAS for the first time, or an experienced SAS programmer wanting a quick start with the new versions, this paper is for you.

Version what?

Even though Version 7 was released less than a year ago, Version 8 is already on its way and will be released soon. This situation makes it hard to talk about one version without including the other. While Version 7 represents a major change from Version 6, Version 8 is described by SAS Institute as introducing primarily performance enhancements. In addition, products (such as Enterprise Miner) that weren't part of Version 7, will be available in Version 8.

Most of the features discussed in this paper were introduced with Version 7. A few are new with Version 8. For features that are new in Version 8, we state that fact. If no mention is made of when a feature was introduced, you can assume it was new in Version 7.

New terminology—old ideas

SAS windowing environment Version 7 brought with it some new terminology. For example, you won't find the term Display Manager, even though the Program Editor, Log, and Output windows are still there . Along with the new Results and Explorer windows, they are now called the SAS windowing environment.

Relational terms SAS has always had many of the features of a relational database. Reflecting that fact, standard relational terms are now being used to describe SAS data sets as shown in Table 1.

Table 1. SAS terminology.

Traditional term	Relational term
data set	table
variable	column
observation	row

This is a sensible change reflecting the fact that SAS is a relational database. However, we do not think that the old SAS terminology will disappear as some people have suggested, because, frankly, it is impossible to describe many common tasks with the relational terminology. For example, if you try to have an intelligible conversation about the TABULATE procedure using only relational terminology you will find yourself spouting terms such as "column columns" and "row columns." The traditional terms "column variables" and "row variables" are more clear. In this paper, we will use both the traditional and relational terminology, depending on which best fits the subject.

SAS OnlineDoc™ CD

Starting with Version 7, the primary source of documentation about the SAS System is electronic. The OnlineDoc is being shipped on a CD with SAS software, and can be accessed directly from the CD-ROM, or through an internal web or file server. The disadvantage of this is that SAS programmers will no longer be able to brag about the number of linear feet of SAS manuals filling their shelves. The advantage is that you can make your computer do the work of searching for your desired topic.

The OnlineDoc includes reference materials for the SAS language, procedures, macros, many SAS products, and host companions. It does not include all books published by SAS Institute such as the Books-by-Users manuals, so save a little room on your book shelf.

Enhanced Editor for Windows

The Enhanced Editor got lots of oohs and aahs when it made its debut at the SUGI 23 opening session in Nashville, Tennessee. However, we have had to wait until Version 8 of the SAS system to get our hands on it. Because it is sensitive to SAS syntax, the Enhanced Editor will make writing bug free SAS code a lot easier . Unfortunately, this editor is currently available only in the Windows operating environment.

Each component of a SAS program is color coded in the Enhanced Editor. For example, SAS keywords are blue, variable names are black, quoted text strings are red, and comments are a nice teal color. And, if you don't like those colors, you can choose your own. The color coding makes it very easy to see when you are missing a closing quote, or missing a semicolon at the end of your statement.

In addition to color coding, the Enhanced Editor organizes your SAS program by visually grouping

statements into the appropriate DATA or PROC step. Grouping is done by either brackets or inserting horizontal lines between steps, or both. It is also possible to collapse steps into a single line so that you can view the overall structure of the program.

Another big difference between the Program Editor and the Enhanced Editor, is that by default, your SAS code does not disappear from the Enhanced Editor window when you submit your program. If you would rather have to recall your program after submitting like you do in the Program Editor, there is an option which will allow you to change the default.

In the Developer's Release of Version 8 of the SAS system, you get both an Enhanced Editor and a Program Editor window by default when you start up SAS.

Long SAS names

The rules for SAS names changed considerably with Version 7 of SAS. Prior to Version 7, SAS names could be only 8 characters long, now many can be longer as shown in Table 2.

Table 2. Maximum lengths for SAS names.

Type of name	Maximum length
array	32
data set member	32
fileref	8
format	8
informat	7
libref	8
macro	32
macro variable	32
variable	32
versioned data set member	28

The other rules for SAS names stay the same as before. SAS names must start with a letter or underscore, and can contain only letters, numerals, and underscores.

We should mention that in Versions 7 and 8 it is possible to use special characters, including spaces, in variable names if you specify the experimental system option VALIDVARNAMES=ANY. But you also have to use name literals of the form `'variable-name'N` for every occurrence of every variable name containing a space or special character. We think that programmers, being inherently lazy, will generally choose to avoid spaces and special characters in variable names.

Mixed case variable names

SAS has always been insensitive to case, and that is still true. You can use upper case, lower case, or mixed case—which ever looks best to you. SAS doesn't care.

The name BIRTHDATE is the same as birthdate, and BirthDate. Now, however, SAS remembers the case of the first occurrence of each variable name and uses that case when printing results. If you use mixed case, then SAS will use the capital letters to decide where to split variable names in headings, a convenient feature that will make your reports prettier.

Example Here is a program containing mixed case variable names.

```
DATA travellog;
   INPUT Destination $8. DepartDate
      DATE8. ReturnDate DATE8.;
   DATALINES;
Belgium 08MAR97 19AUG97
Hawaii  28DEC97 07JUL98
   ;

PROC PRINT DATA = travellog;
   FORMAT DepartDate ReturnDate
      MMDDYY8.;
   TITLE 'Travel Log';
RUN;
```

Here is the output from PROC PRINT containing split variable names.

```
                Travel Log                    1

                          Depart      Return
Obs Destination             Date        Date

 1    Belgium           03/08/97    08/19/97
 2    Hawaii            12/28/97    07/07/98
```

In the past you could use the SPLIT= option in PROC PRINT to get this effect, but now SAS does it automatically—and for all procedures as needed.

Long variable labels

If 32 character variable names isn't long enough for you, you'll be glad to know that variable labels, formerly limited to 40 characters, can now be up to 256 characters long.

Long character values

In Version 6 character values were limited to 200 bytes. Starting with Version 7, character values can be up to 32,767 bytes long.

YEARCUTOFF= default value

With Version 7, the default value for the YEARCUTOFF= system option increased from 1900 to 1920. SAS uses the YEARCUTOFF= system option to determine in which century a two-digit year belongs, so Version 7 interprets two-digit years as occurring between 1920 and 2019. Version 6, by default, interprets the year 00 as 1900; while Versions 7and 8 interpret the year 00 as 2000.

DATASTMTCHK= system option

This system option, new with Version 7, controls whether keywords can be used as data set names. Of course, very few people want to use keywords as data set names, but they do it accidentally when they forget to put a semicolon at the end of a DATA statement. This problem can cause SAS to overwrite an existing data set with bad data.

With the default value of the DATASTMTCHK= system option, COREKEYWORDS, SAS will not allow the words MERGE, RETAIN, SET, or UPDATE as data set names. However, you can still run into problems if your DATA statement is followed by some other statement such as INFILE, LENGTH, or ATTRIB. You can prevent any SAS keyword from being used as a data set name by setting the value of this option to ALLKEYWORDS in an OPTIONS statement.

For example, the following log shows how the ALLKEYWORDS option prevents SAS from creating a data set named INFILE when the semicolon is missing from a DATA statement.

```
1   OPTIONS DATASTMTCHK = ALLKEYWORDS;
2
3   DATA travellog
4     INFILE 'c:\MySASLib\travellog.dat';
      ------
      57
ERROR 57-185: INFILE is not allowed in
the DATA statement when option
DATASTMTCHK=ALLKEYWORDS.  Check for a
missing semicolon in the DATA statement,
or use DATASTMTCHK=NONE.

5   INPUT Destination $8. DepartDate
6      DATE8. ReturnDate DATE8.;
7   RUN;
```

MERGENOBY= system option

Introduced with Version 8, this system option tells SAS to issue a warning or error message whenever data sets are merged without a BY statement. By default, SAS issues no warning, and performs an unmatched merge which nearly always produces garbage. You can change the setting of the MERGENOBY system option with this statement:

```
OPTIONS MERGENOBY = WARN;
```

SAS will still perform the unmatched merge, but will also print this message in your log:

```
WARNING: No BY statement was
specified for a MERGE statement.
```

To prevent an unmatched merge entirely, set the MERGENOBY option to ERROR like this:

```
OPTIONS MERGENOBY = ERROR;
```

Then if you attempt to merge data sets without a BY statement, SAS will stop executing the DATA step and print this message in your log:

```
ERROR: No BY statement was
specified for a MERGE statement.
```

REPEMPTY= data set option

Another feature introduced with Version 8 is the REPEMPTY= data set option. By setting this option to NO, you can prevent an existing SAS data set from being replaced with an empty SAS data set. For example, in the DATA step below, if the data set NEWTRAVEL is empty for some reason, by default SAS will replace the TRAVELLOG data set with an empty data set.

```
DATA travellog;
   SET newtravel;
   RUN;
```

You can prevent this with the REPEMPTY= data set option.

```
DATA travellog (REPEMPTY = NO);
   SET newtravel;
   RUN;
```

If you run this when the NEWTRAVEL data set is empty, SAS will print this message in your log.

```
WARNING: Data set WORK.NEW was not
   replaced because REPEMPTY=NO and
   the replacement file is empty.
```

Transporting SAS data sets

Portability is nothing new for SAS data sets, that is you could write a SAS data set in one operating environment and read it on a another. But doing this was usually a challenge. Starting with Version 7, this formerly formidable task has become in comparison almost trivial.

In most cases you no longer need to create a transport data set. Instead you can simply read Version 7 or 8 SAS data sets created in other operating environments without conversion (except for CMS and OS/390 bound libraries). You can directly access SAS data sets created in another operating environment, but residing on your shared network. Or, you can transfer SAS data sets between operating environments using binary transfer.

It takes more resources to read a data set created in a different operating environment, so, if you wish, you can even write SAS data sets in the format of an operating system other than your own using Cross Environment Data Access (CEDA).

Directly referencing permanent SAS data sets

An important change that will make SAS easier for beginners is the ability to refer directly to permanent SAS data sets. Starting with Version 7 you can, in many cases, ignore those elusive librefs. Just put your operating environment's name for your data set between quotes and insert it in your program. For example, in the Windows operating environment you could create a SAS

data set named MUSEUMS in a directory named MySASLib on your C drive with statements like this:

```
DATA 'c:\MySASLib\museums';
   INFILE 'c:\MyRawData\tour.dat';
   INPUT Name $ City $ Price;
   RUN;
```

If you run this, your log will contain a note like this:

```
NOTE: The data set c:\MySASLib\museums has
   10 observations and 3 variables.
```

Notice that the data set name, in the program and in the log, does not contain an extension. SAS automatically appends the appropriate file extension. If you list the files in this directory, MySASLib, you will see one named `museums.sas7bdat`. SAS uses the same file extension, `sas7bdat`, for SAS data sets in both Version 7 and 8.

Suppose the next day you want to use this data set again. You can refer to it by using the same name you used to create it.

```
PROC PRINT
   DATA = 'c:\MySASLib\museums';
   RUN;
```

Here is the general form of the DATA statement for creating permanent SAS data sets under different operating environments:

Windows, OS/2:	`DATA 'drive:\directory\filename';`
UNIX:	`DATA '/home/path/filename';`
OpenVMS, VMS:	`DATA '[userid.directory]filename';`
CMS:	`DATA 'filename filetype filemode';`
OS/390:	`DATA 'OS.data.set.name';`

For directory-based operating environments, if you leave out the directory or path, then SAS uses the current working directory. For example, this statement would create a permanent SAS data set named MUSEUMS in your current working directory.

```
DATA 'museums';
```

Under Windows the name of the current working directory is displayed in the lower-right corner. You can change the directory temporarily by double-clicking on the directory name which will open the Change Folder window.

Since librefs are now optional, there has been a shift in the terminology used in SAS documentation. You will often see the term data set or table used to describe the member name, as if librefs no longer existed. In fact, the libref is still there even with direct referencing. Since you haven't specified a libref, SAS figures out the libref for you. If you are using a data set in a library that already has a libref, SAS will figure that out and use the existing libref. If no libref exists then SAS will make one up, such as Wc000001. You don't need to know the name of the libref that SAS makes up, but experienced SAS users may be curious about it. You can see it in the SAS windowing environment by issuing the command LIBNAME to open the LIBNAME window.

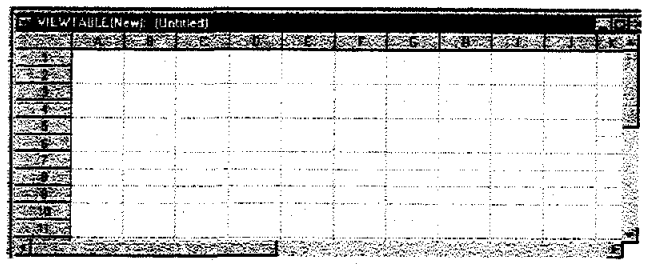
Unfortunately, a few features in Versions 7 and 8 (including the Viewtable editor, and the IMPORT and EXPORT procedures) do not yet accept direct referencing.

Viewtable window

The Viewtable window is an easy way to create new tables (also called data sets), or browse and edit existing tables. True to its name, the Viewtable window displays data in a tabular format. While not a spreadsheet (because the cells are not independent), you may find it helpful to think of Viewtable as similar to a spreadsheet.

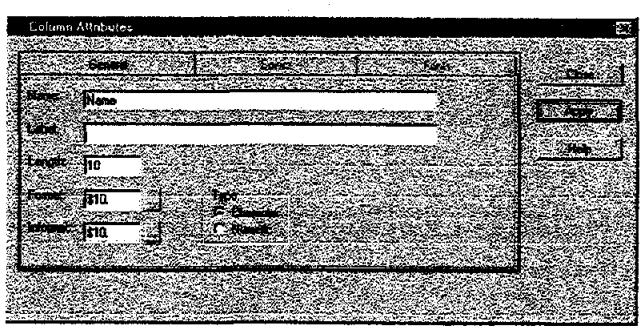
Viewtable was introduced with release 6.12, but in that release you needed to have SAS/FSP, which is licensed separately, to be able to edit or sort a table. Editing is so critical that Viewtable was fairly useless for people who didn't have SAS/FSP. Now in Version 7, all features of Viewtable are available with Base SAS with one exception. If you are using a non-graphical monitor, then SAS uses FSVIEW to display your tables, so you still need SAS/FSP.

To open the Viewtable window, select `Table Editor` from the `Tools` menu. (In release 6.12 select `Globals-Manage-Open Table`.) An empty Viewtable window will appear.



This table contains no data. Instead you see rows (or observations) labeled with numbers and columns (or variables) labeled with letters. You can start typing data into this default table, and SAS will automatically figure out if your columns are numeric or character, and assign informats and formats. However, it's a good idea to tell SAS about your data so each column is set up the way you want. You do this with the Column Attributes window.

Column Attributes window The letters at the tops of columns are default variable names. By double-clicking on a letter, you can open a Column Attributes window for that column.



This window contains default values which you can replace with the values you desire. When you are satisfied with the values, click on `Apply`. To switch to a new column click on that column in the Viewtable window. When you are finished changing column attributes click on `Close`.

Entering data Once you have defined your columns you are ready to type in your data. To move the cursor, click on a field, or use the tab key. Here is a table with column attributes defined and data entered.

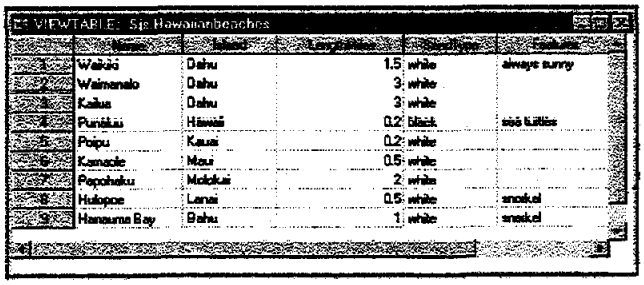
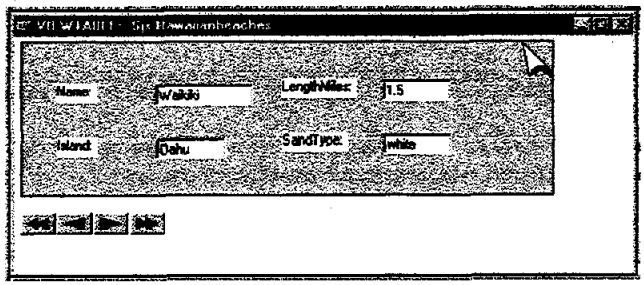
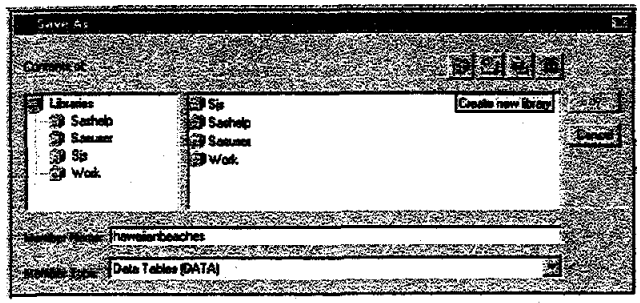


Table versus Form View The screen above shows the data in Table View. Alternatively, you may display your data one row at a time, called Form View. To do this click on the Form View icon or select `Form View` from the `View` menu. Here is Form View for the beaches data.



Saving your table To save a table, select `Save As...` from the `File` menu. Select a library, and then specify the member name of your table. The libraries displayed correspond to locations (such as directories) on your computer. If you want to save your table in a different location, you can add another library by clicking on the New Library icon. Type in a name for the new library and its path. Then click on `OK`. You can specify the member name either by typing in the Member Name field, or by clicking on the library and then clicking on the icon of an existing table. When you have specified the library and member name, click on `Save`.



Opening and editing an existing table To browse or edit an existing table, first select `Table Editor` from the `Tools` menu to open the Viewtable window. Then select `Open` from the `File` menu. Click on the library you want and then on the table name. If the table you want to open is not in any of the existing libraries, click on the New Library icon. Type in a name for the new library and its path. Then click on `OK`.

Your table opens in browse mode. To edit the table select `Edit Mode` from the `Edit` menu. You may then edit fields by clicking on them, or add and delete rows by selecting these from the Edit menu.

Sorting You can sort your data by clicking on a column and then clicking on the icon for Sort Ascending or Sort Descending, or, for more complex sorts, selecting `Sort` from the `Data` menu to open the Sort window.

Using your table in a SAS program Tables that you create in Viewtable can be used in programs just as tables created in programs can be used in Viewtable. For example, if you saved your table in the SASUSER library and named it HAWAIIANBEACHES, you could print it with this program:

```
PROC PRINT
   DATA = sasuser.hawaiianbeaches;
RUN;
```

For more detailed information about Viewtable we recommend the book *Window by Window: Capture Your Data Using the SAS System*, Chapter 9, Data Entry Using Viewtable.

IMPORT and EXPORT procedures

The IMPORT and EXPORT wizards were first introduced with release 6.12. If you've never tried them, you should. These easy-to-use, GUIs (Graphical User Interfaces) lead you through a series of windows and then convert delimited files to SAS data sets, or vice versa. To access the wizards you simply select `Import` or `Export` from the `File` menu in the SAS windowing environment. The only problem with the wizards is that you can't put them in a program. That problem is gone with Version 7—for OS/2, UNIX, VMS, and Windows—thanks to PROC IMPORT and PROC EXPORT.

Of course, you can still write a DATA step to read or write a delimited data file, but PROC IMPORT and EXPORT are often easier. In fact, if you look in your SAS log you will see that PROC IMPORT and EXPORT write DATA steps for you.

IMPORT will:

- scan your data file and automatically determine the variable types (character or numeric)
- assign proper lengths to character variables
- recognize some date formats
- treat two consecutive delimiters in your data file as a missing value
- read values enclosed by quotes
- assign missing values to variables when it runs out of data on a line
- use the first line of your file for variable names if it contains column headers.

EXPORT does basically the same thing in reverse.

If you have SAS/ACCESS for PC File Formats (which is licensed separately) you can also use PROC IMPORT and EXPORT (and the wizards) to read and write data for dBase, Lotus, Excel, and Microsoft Access. In this paper we describe the features included with Base SAS, but the procedures work basically the same for database and spreadsheet files. For more information about importing and exporting PC database and spreadsheet files, see section 2.18 of *The Little SAS Book, Second Edition*, or Delwiche and Slaughter (1998) "The Programmer's Solution to the Import/Export Wizard."

Using PROC IMPORT The simplest form of the IMPORT procedure is:

```
PROC IMPORT DATAFILE = 'filename'
   OUT = data-set;
```

where the file you want to read follows the DATAFILE= option and name of the SAS data set you want to create follows the OUT= option. SAS will determine the type of file by the extension of the filename as shown in Table 3.

Table 3. Data file types, extensions, and DBMS identifiers for delimited files in PROC IMPORT.

Type of File	Extension	DBMS Identifier
Comma-delimited	.csv	CSV
Tab-delimited	.txt	TAB
Other delimiters		DLM

If your file does not have the proper extension, or your file has delimiters other than commas or tabs, then you must specify a DBMS identifier using the DBMS= option in the PROC IMPORT statement.

You may also want to use the REPLACE option. Without the REPLACE option SAS will not overwrite a data set with the same name.

The IMPORT procedure will, by default, get variable names from the first line in your data file. If the first line in your data file does not contain variable names, then add the GETNAMES=NO; statement. When you use this statement, IMPORT will assign the variable names VAR1, VAR2, VAR3, and so on.

If your data file is of the type DLM, IMPORT assumes that the delimiter is a space. If you have some other delimiter, then specify it in the DELIMITER= statement.

Here is the IMPORT procedure with all its options and optional statements.

```
PROC IMPORT DATAFILE = 'filename'
      OUT = data-set
      DBMS = identifier
      REPLACE;
   GETNAMES = NO;
   DELIMITER = 'delimiter-character';
```

Example Table 4 shows a comma separated values (CSV) file. The file gives information about several tourist attractions: country, state or province, name of the attraction, adult price, currency, and the date the information was last updated. The first line of the file has labels that can be used for variable names. The file has some missing data in the Price field (indicated by two consecutive commas), and a date field, named Updated. One value for Attraction is enclosed in quotes because it has an embedded comma.

The following PROC IMPORT reads the data, using only a single statement, and creates a SAS data set named TOURISTSPOTS. The data set, printed by PROC PRINT, appears in Table 5.

```
PROC IMPORT DATAFILE =
   'c:\MyRawData\touristspots.csv'
   OUT=touristspots
   REPLACE;
PROC PRINT;
   TITLE 'Tourist Attractions';
RUN;
```

Notice that, in addition to correctly identifying variable types and lengths, IMPORT also assigns a date informat and format to the variable, Updated, and takes the variables' names from the first line in the data file. For fields that have spaces in the variable names, SAS inserts an underscore, so that the variable still conforms to the default rules for variable names.

Table 4. CSV data file.

```
Country,State or Province,Attraction,Price,Currency,Updated
USA,Hawaii,Waikiki Aquarium,6,Dollars,07/11/98
USA,Hawaii,Polynesian Cultural Center,75,Dollars,07/11/98
USA,Hawaii,Punaluu Black Sand Beach,0,Dollars,07/11/98
USA,Hawaii,Hawaii Volcanoes National Park,0,Dollars,07/11/98
USA,Hawaii,Koko Crater Botanical Garden,0,Dollars,07/11/98
Belgium,Brussel,Royal Army Museum,0,BEF,03/06/97
Belgium,Brussel,"Atomium,Mini-Europe",500,BEF,03/06/97
Belgium,Luxembourg,Bastonge American Memorial,245,BEF,03/06/97
Belgium,Liege,Caves of Remouchamps,290,BEF,03/06/97
Belgium,Brabant,Grottes De Folx-les-Caves,,BEF,03/06/97
```

Table 5. SAS data set created from a CSV file.

Tourist Attractions 1

Obs	Country	State_or_ Province	Attraction	Price	Currency	Updated
1	USA	Hawaii	Waikiki Aquarium	6	Dollars	07/11/1998
2	USA	Hawaii	Polynesian Cultural Center	75	Dollars	07/11/1998
3	USA	Hawaii	Punaluu Black Sand Beach	0	Dollars	07/11/1998
4	USA	Hawaii	Hawaii Volcanoes National Park	0	Dollars	07/11/1998
5	USA	Hawaii	Koko Crater Botanical Garden	0	Dollars	07/11/1998
6	Belgium	Brussel	Royal Army Museum	0	BEF	03/06/1997
7	Belgium	Brussel	Atomium,Mini-Europe	500	BEF	03/06/1997
8	Belgium	Luxembourg	Bastonge American Memorial	245	BEF	03/06/1997
9	Belgium	Liege	Caves of Remouchamps	290	BEF	03/06/1997
10	Belgium	Brabant	Grottes De Folx-les-Caves	.	BEF	03/06/1997

Using PROC EXPORT The EXPORT procedure does basically the opposite of the IMPORT procedure. With EXPORT you can convert your SAS data sets into delimited files, and, if you have SAS/ACCESS for PC File Formats, you can also convert data sets into PC spreadsheet and database files.

The general form of PROC EXPORT is:

```
PROC EXPORT DATA = data-set
            OUTFILE = 'filename';
```

where *data-set* is the SAS data set you are reading, and *filename* is a name you make up for the output data file. As with PROC IMPORT, SAS uses the last part of the filename, the extension, to decide what type of file to create. You can also specify the file type by adding the DBMS= option. Table 6 shows the filename extensions and DBMS identifiers currently available with Base SAS.

Table 6. Data file types, extensions, and DBMS identifiers for delimited files in PROC EXPORT.

Type of File	Extension	DBMS Identifier
Comma-delimited	.csv	CSV
Tab-delimited	.txt	TAB
Other delimiters		DLM

Notice that for space-delimited files, there is no standard extension so you must use the DBMS= option. The REPLACE option tells SAS that it is ok to replace any existing file with the same name. If you want to use a delimiter other than tabs, spaces, or commas, then add the DELIMITER= statement. Here is the EXPORT procedure with all its options and optional statements.

```
. PROC EXPORT DATA = data-set
        OUTFILE = 'filename'
        DBMS = identifier
        REPLACE;
    DELIMITER = 'delimiter-character';
```

Example Suppose you wanted to read the touristspots data into a word processor. A good way to do this is with a tab-delimited file. You can create a tab-delimited file using PROC EXPORT with just a single statement.

```
PROC EXPORT DATA = touristspots
   OUTFILE =
      'c:\MyRawData\touristspots.txt'
   REPLACE;
RUN;
```

This program creates a file named TOURISTSPOTS.TXT in the MyRawData directory. Because the file has a .txt extension, SAS knows that it should be tab-delimited. Table 7 shows what this file looks like after being opened in a word processor and having the tabs set.

FILE statement with DLM= and DSD options

The DLM= (or DELIMITER=) and DSD (Delimiter Separated Data) options are not new. They are new, however, in the FILE statement. In the past, you could use these options for reading delimited raw data (such as comma-separated values) into SAS with an INFILE statement. Starting with Version 7, you can use these options in reverse to write delimited data.

These options are more difficult to use than PROC EXPORT or the EXPORT wizard, but there are still reasons to use this method. First, if you are working in the OS/390 or CMS operating environments, PROC EXPORT and its wizard are not available. Second, with these options and the flexibility of PUT statements, you can create custom files that you couldn't get from EXPORT.

The general way to use these options is:

```
DATA _NULL_;
   SET data-set-name;
   FILE 'filename' DSD DLM = ',' ;
   PUT _ALL_;
```

This creates a raw data set containing all your variables separated by commas. Two consecutive commas indicate missing data.

Output Delivery System

Probably the single biggest innovation introduced with Version 7 is the Output Delivery System (ODS). Before ODS every procedure handled its own output, whether that output was a report or a data set. Now every procedure passes its output to the Output Delivery System which decides where the output will go and what it will look like when it gets there.

The default destination for output is the standard listing that goes in your Output window. In addition to the listing file, you can tell SAS to put your output in a SAS data set, or in Hyper Text Markup Language (HTML) format. If you have Version 8 of the SAS System, you can also send your output to the PRINTER destination (which is experimental in Version 7). The PRINTER destination is for high resolution printers, and can create a printable file (such as a postscript file) or send output automatically to your printer. Other output destinations, including Rich Text Format (RTF), are expected to be added soon, so check your release.

The Output window, in the SAS windowing environment, looks pretty much the same as in previous releases, but it's really quite different. Along with the Output window, you now have a Results window which displays a tree structure for listing output. This Results window is like a table of contents for your output. If you double click on a piece of output, such as a PROC PRINT, then that report will pop to the top of your Output window providing an easy way to navigate your results.

The Results window is intuitive and easy to use. Unfortunately, other output destinations are not so easy to use. The statements can be confusing and complicated. You sometimes have to be very particular about where you put the ODS statements. We will show you one alternative output destination: creating SAS data sets from procedure output. For an example of creating ODS HTML output, see section 4.15 of *The Little SAS Book, Second Edition*.

Table 7. Tab-delimited file created by PROC EXPORT, read into a word processor with tabs set.

Country	State_or_Province	Attraction	Price	Currency	Updated
USA	Hawaii	Waikiki Aquarium	6	Dollars	07/11/1998
USA	Hawaii	Polynesian Cultural Center	75	Dollars	07/11/1998
USA	Hawaii	Punaluu Black Sand Beach	0	Dollars	07/11/1998
USA	Hawaii	Hawaii Volcanoes National Park	0	Dollars	07/11/1998
USA	Hawaii	Koko Crater Botanical Garden	0	Dollars	07/11/1998
Belgium	Brussel	Royal Army Museum	0	BEF	03/06/1997
Belgium	Brussel	Atomium,Mini-Europe	500	BEF	03/06/1997
Belgium	Luxembourg	Bastonge American Memorial	245	BEF	03/06/1997
Belgium	Liege	Caves of Remouchamps	290	BEF	03/06/1997
Belgium	Brabant	Grottes De Folx-les-Caves	.	BEF	03/06/1997

ODS TRACE With ODS every result produced by a procedure has a name. This gives you a handle, a way to access, each piece of information generated by the procedure. Once you know the name of the piece of output you want, you can save it. You can search through the documentation to find out the names of the output for a particular procedure, but there is an easier way—use the ODS TRACE statement in your program.

There are two ODS TRACE statements: one to turn on the trace (ODS TRACE ON), and one to turn it off (ODS TRACE OFF). When the trace is turned on, information about your procedure output, including names of the output parts, is written to the SAS log. To use these statements, insert them on either side of the procedure whose output you want to capture like this:

```
ODS TRACE ON;

your program statements here;

ODS TRACE OFF;
```

Example The following statements use the beaches data shown earlier to produce a report from PROC TABULATE. To find out the name of the results, the ODS TRACE ON and ODS TRACE OFF statements have been inserted before and after the procedure.

```
ODS TRACE ON;
PROC TABULATE
   DATA= 'c:\MySASLib\hawaiianbeaches';
   CLASS SandType;
   VAR LengthMiles;
   TABLE SandType, MAX * LengthMiles N;
RUN;
ODS TRACE OFF;
```

Here is an excerpt from the SAS log showing the results of the ODS trace.

```
Output Added:
-------------
Name:         Report
Label:        Cross-tabular summary report
Data Name:    Summary
Data Label:   Summary statistics
Path:         Tabulate.Report
-------------
```

The TABULATE procedure produces one output part named Report. Some procedures produce several pieces of output, each with a different name.

ODS OUTPUT Once you know the name of the piece of output that you want, you can use the ODS OUTPUT statement to save the results in a SAS data set. The simplest form of the ODS OUTPUT statement is:

```
ODS OUTPUT name = data-set;
```

where name is the name (or path) of the output you want to save, and data-set is a name you make up for the new SAS data set.

You need to be careful where you put the ODS OUTPUT statement in your program. This statement executes immediately, and the data set opened by the ODS OUTPUT statement remains open until the next encounter of an end of a PROC step. Because of this, the statement will apply to whatever PROC is currently being processed or the next PROC if there is not a current PROC. To be sure that you get the right output, put the ODS statement after your PROC statement, and before the next PROC, DATA, or RUN statement.

Example Now that you know the name of PROC TABULATE's output you can rerun the program capturing the output in a data set.

```
PROC TABULATE DATA = 'hawaiianbeaches';
   CLASS SandType;
   VAR LengthMiles;
   TABLE SandType, MAX * LengthMiles N;
ODS OUTPUT REPORT = sandoutdata;

PROC PRINT DATA = sandoutdata;
   TITLE 'ODS Output Data Set';
RUN;
```

Notice that the ODS TRACE statements have been removed, and the ODS OUTPUT statement inserted. The ODS OUTPUT statement tells SAS to put the REPORT output in a data set named SANDOUTDATA. Then the output data set is printed using PROC PRINT as shown in Table 8.

Table 8. Output showing data set captured using ODS.

ODS Output Data Set 1

Obs	Sand Type	_TYPE_	_PAGE_	_TABLE_	Length Miles_ Max	N
1	black	1	1	1	0.2	1
2	rocky	1	1	1	1.0	2
3	white	1	1	1	3.0	9

CONCLUSIONS

The new versions of SAS are easy to get used to. When Lora received her beta copy of Version 7, she installed it on her home computer. It was so easy to get used to that the next time she used SAS at work, she automatically typed in long variable names. Of course, her program failed because she was using Version 6 at work. Even after hearing Lora's story, Susan did exactly the same thing the first time she went back to Version 6.

We think that many of the changes in Versions 7 and 8 are like that. They are so easy to get used to that you may feel there hasn't been much of a change, when, in fact, if you go back to Version 6 you will see that the new versions really have changed the way you use SAS everyday.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

References

Delwiche, Lora D. and Susan J. Slaughter (1998). *The Little SAS Book: A Primer, Second Edition.* SAS Institute, Cary, NC.

Delwiche, Lora D. and Susan J. Slaughter (1998). The Programmer's Solution to the Import/Export Wizard. WUSS '98 Proceedings. SAS Institute, Cary, NC.

SAS OnlineDoc, Version 7.1 (1999). SAS Institute, Cary, NC.

Window by Window: Capture Your Data Using the SAS System (1997). SAS Institute, Cary, NC.

About the Authors

Lora Delwiche and Susan Slaughter are authors of *The Little SAS Book: A Primer* published by SAS Institute. They recently completed a second edition of *The Little SAS Book* for the new versions. They may be contacted at:

Susan J. Slaughter (530) 756-8434
sjslaughter@mother.com

Lora D. Delwiche (530) 752-6285
lddelwiche@ucdavis.edu