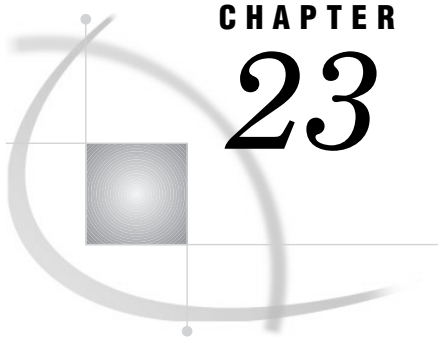

CHAPTER

23

The IMPORT Procedure

Overview

The IMPORT procedure reads data from an external data source and writes it to a SAS data set. External data sources can include DBMS tables, PC files, spreadsheets, and delimited external files (which are files containing columns of data values that are separated by a delimiter such as a blank or a comma).

Once you invoke PROC IMPORT, the procedure reads the input file and writes the data to a SAS data set, with the SAS variables defined based on the input records. PROC IMPORT imports the data by one of the following methods:

- generated DATA step code
- generated SAS/ACCESS code
- translation engines.

You control the results with options and statements that are specific to the input data source. PROC IMPORT produces the specified output SAS data set and writes information regarding the import to the SAS log. In the log, you see the DATA step or the SAS/ACCESS code generated by PROC IMPORT. If a translation engine is used, then no code is submitted.

PROC IMPORT is available on the following hosts:

- OS/2
- UNIX
- OpenVMS
- Windows 95
- Windows NT
- Windows 98

Procedure Syntax

PROC IMPORT

DATAFILE=*"filename"* | TABLE=*"tablename"*
OUT=*SAS-data-set*
<DBMS=*identifier*><REPLACE>;
<*data-source-statements*>;

PROC IMPORT Statement

Featured in: All examples

PROC IMPORT
DATAFILE=*"filename"* | TABLE=*"tablename"*
OUT=*SAS-data-set*
<DBMS=*identifier*><REPLACE>;

Required Arguments

DATAFILE="*filename*"
specifies the complete path and filename of the input PC file, spreadsheet, or delimited external file. If the name does not include special characters (such as the backslash in a path), lowercase characters, or spaces, you can omit the quotes.

Featured in: Example 1 on page 617 and Example 2 on page 620

TABLE="*tablename*"
specifies the table name of the input DBMS table. If the name does not include special characters (such as question marks), lowercase characters, or spaces, you can omit the quotes. Note that the DBMS table name may be case-sensitive.

Requirement: When you import a DBMS table, you must specify the DBMS= option.

Featured in: Example 3 on page 621

OUT=*SAS-data-set*
identifies the output SAS data set with either a one- or two-level SAS name (library and member name). If the specified SAS data set does not exist, PROC IMPORT creates it. If you specify a one-level name, PROC IMPORT uses the WORK library.

Featured in: All examples

Options

DBMS=*identifier*
specifies the type of data to import. For example, DBMS=DBF specifies to import a dBASE file. For PC files, spreadsheets, and delimited external files, you do not have to specify DBMS= if the filename specified with DATAFILE= contains a valid extension so that PROC IMPORT can recognize the type of data. For example, PROC IMPORT recognizes the filename ACCOUNTS.WK1 as a Lotus 1 spreadsheet and the filename MYDATA.CSV as a delimited external file that contains comma-separated data values. PROC IMPORT can recognize the difference between Excel Version 4 and Version 5 spreadsheets when you use the extension .XLS, regardless of whether

you specify DBMS=EXCEL, DBMS=EXCEL4, or DBMS=EXCEL5. However, you must specify DBMS=EXCEL97 to import Excel 97 spreadsheets. If you do not specify an identifier or if the extension of the filename is not recognized, an error is returned.

To import a DBMS table, you must specify DBMS= using a valid database product. For example, DBMS=ACCESS imports a Microsoft Access table.

The DBMS= specification can include the values listed in the following table:

Identifier	Input Data Source	Extension
ACCESS	Microsoft Access database	.MDB
DBF	dBASE file	.DBF
WK1	Lotus 1 spreadsheet	.WK1
WK3	Lotus 3 spreadsheet	.WK3
WK4	Lotus 4 spreadsheet	.WK4
EXCEL	Excel Version 4 or 5 spreadsheet	.XLS
EXCEL4	Excel Version 4 spreadsheet	.XLS
EXCEL5	Excel Version 5 spreadsheet	.XLS
EXCEL97	Excel 97 spreadsheet	.XLS
DLM	delimited file (default delimiter is a blank)	.*
CSV	delimited file (comma-separated values)	.CSV
TAB	delimited file (tab-delimited values)	.TXT

Restriction: The data sources available to you depend on the SAS/ACCESS products that you have licensed. If you do not have any SAS/ACCESS products licensed, then the only types of data source files available to you are .CSV, .TXT, and delimited files.

Restriction: The OS/2 operating environment does not support Excel 5 and Excel 97 spreadsheets.

Featured in: Example 1 on page 617 and Example 3 on page 621

REPLACE
overwrites an existing SAS data set. If you do not specify REPLACE, PROC IMPORT does not overwrite an existing data set.

Featured in: Example 1 on page 617

Data Source Statements

Featured in: All examples

PROC IMPORT provides a variety of statements that are specific to the input data source.

Statements for PC Files, Spreadsheets, or Delimited External Files

Table 23.1 on page 614 describes which statements are available to import PC files, spreadsheets, and delimited external files, and it denotes which statements are valid for a specific data source. For example, Excel spreadsheets have optional statements to indicate whether column names are in the first row of data or which sheet and range of data to import, while a dBASE file (.DBF) does not. For more information about PC file formats, see *SAS/ACCESS Software for PC File Formats: Reference*.

Table 23.1 Statements for PC Files, Spreadsheets, and Delimited External Files

	Statements					
Input Type	GETNAMES=	RANGE=	SHEET=	DELIMITER=	GETDELETED=	DATAROW=
DBF					X	
WK1	X	X	X			
WK3	X	X	X			
WK4	X	X	X			
EXCEL	X	X	X			
EXCEL4	X	X	X			
EXCEL5	X	X	X			
EXCEL97	X	X	X			
DLM	X			X		X
CSV	X					X
TAB	X					X

DATAROW=*n*
starts reading data from row number *n* in the external file.

Default:

1 when GETNAMES=NO

2 when GETNAMES=YES (default for GETNAMES=)

Interaction: When GETNAMES=YES, DATAROW must be ≥ 2. When GETNAMES=NO, DATAROW must be ≥ 1.

DELIMITER=*'char'|'nn'x*
for a delimited external file, specifies the delimiter that separates columns of data in the input file. You can specify the delimiter as a single character or as a hexidecimal value. For example, if columns of data are separated by an ampersand, specify DELIMITER='&'. If you do not specify DELIMITER=, PROC IMPORT assumes that the delimiter is the blank. You may replace the equals sign with a blank.

Featured in: Example 1 on page 617

GETDELETED=YES|NO
for a DBF file, indicates whether to write records to the SAS data set that are marked for deletion but have not been purged. You may replace the equals sign with a blank.

Default: NO

GETNAMES=YES|NO
for spreadsheets and delimited external files, determines whether to generate SAS variable names from the column names in the input file's first row of data. If you specify GETNAMES=NO or if the column names are not valid SAS names, PROC IMPORT uses the variable names VAR0, VAR1, VAR2, and so on. You may replace the equals sign with a blank.

Default: YES

Featured in: Example 1 on page 617 and Example 2 on page 620

RANGE=*range-name*|*absolute-range*
subsets a spreadsheet by identifying the rectangular set of cells to import from the specified spreadsheet. The syntax for range-name and absolute-range is native to the file being read. The *range-name* is the name that is assigned to a range address within a spreadsheet. The *absolute-range* identifies the top left cell that begins the range and bottom right cell that ends the range. The beginning and ending cells are separated by two periods. For example, C9..F12 specifies a cell range that begins at cell C9, ends at cell F12, and includes all the cells in between. If you do not specify RANGE=, PROC IMPORT reads the entire spreadsheet. You may replace the equals sign with a blank.

Restriction: You cannot use *absolute-range* with Excel 97 spreadsheets.

SHEET=*spreadsheet-name*
for spreadsheets, identifies a particular spreadsheet to read from a group of spreadsheets, for example, SHEET=PRICES. Use this statement with spreadsheet files that support multiple spreadsheets within a single file, such as EXCEL5, EXCEL97, WK3, and WK4. The naming convention for the spreadsheet name is native to the file being read. If you do not specify SHEET=, PROC IMPORT reads the first spreadsheet in the file. You may replace the equals sign with a blank.

Featured in: Example 2 on page 620

Statements for DBMS Tables

The following data source statements are available to establish a connection to the DBMS when you import a DBMS table.

DATABASE="*database*"
specifies the complete path and filename of the database that contains the specified DBMS table. If the database name does not contain lowercase characters, special characters, or national characters, you can omit the quotes. You may replace the equals sign with a blank.

Note: A default may be configured in the DBMS client software; however, the SAS System does not generate a default value. △

Featured in: Example 3 on page 621

DBPWD="*database password*"
specifies a password that allows access to a database. You may replace the equals sign with a blank.

Interaction: DBPWD= cannot be used with PWD=.

Featured in: Example 3 on page 621

MEMOSIZE=*field-length*
specifies the field length for importing Microsoft Access Memo fields.

Range: 255–32,767

Default: 1024

Tip: To prevent Memo fields from being imported, you can specify MEMOSIZE=0.

PWD=*"password"*
specifies the user password used by the DBMS to validate a specific userid. If the password does not contain lowercase characters, special characters, or national characters, you can omit the quotes. You may replace the equals sign with a blank.

Note: The DBMS client software may default to the userid and password that was used to log in to the operating environment; the SAS System does not generate a default value. △

Interaction: PWD= cannot be used with DBPWD=.

Featured in: Example 3 on page 621

UID=*"userid"*
identifies the user to the DBMS. If the userid does not contain lowercase characters, special characters, or national characters, you can omit the quotes. You may replace the equals sign with a blank.

Note: The DBMS client software may default to the userid and password that was used to log in to the operating environment; the SAS System does not generate a default value. △

Featured in: Example 3 on page 621

WGDB=*"workgroup-database-name"*
specifies the workgroup (security) database name that contains the USERID and PWD data for the DBMS. If the workgroup database name does not contain lowercase characters, special characters, or national characters, you can omit the quotes. You may replace the equals sign with a blank.

Note: A default workgroup database may be used by the DBMS; the SAS System does not generate a default value. △

Featured in: Example 3 on page 621

Security Levels for Microsoft Access Tables

Microsoft Access tables have three levels of security, for which specific combinations of security statements must be used.

None
Do not specify DBPWD=, PWD=, UID=, or WGDB=.

Password
Specify only DBPWD=.

User-level
Specify only PWD=, UID=, and WGDB=.

Each statement has a default value; however, you may find it necessary to provide a value for each statement explicitly.

Examples

Example 1: Importing a Delimited External File

Procedure features:

PROC IMPORT statement arguments:

DATAFILE=
DBMS=
OUT=
REPLACE

Data source statements:

DELIMITER=
GETNAMES=

Other features:

PRINT procedure

This example imports the following delimited external file and creates a temporary SAS data set named WORK.MYDATA.

```
Region&State&Month&Expenses&Revenue
Southern&GA&JAN97&2000&8000
Southern&GA&FEB97&1200&6000
Southern&FL&FEB97&8500&11000
Northern&NY&FEB97&3000&4000
Northern&NY&MAR97&6000&5000
Southern&FL&MAR97&9800&13500
Northern&MA&MAR97&1500&1000
```

Program

DATAFILE= specifies the input file. The filename does not contain an extension.

```
proc import datafile="/myfiles/mydata"
```

OUT= identifies MYDATA as the output SAS data set.

```
            out=mydata
```

DBMS= specifies that the input file is a delimited external file.

```
            dbms=dlm
```

REPLACE specifies to overwrite the data set if it exists.

```
            replace;
```

DELIMITER= specifies that the delimiter that separates the columns of data in the input file is the ampersand.

```
    delimiter='&';
```

GETNAMES= specifies that PROC IMPORT generates the variable names from the first row of data in the input file.

```
    getnames=yes;
run;
```

The PRINT procedure displays the output data set, WORK.MYDATA.

```
options nodate ps=60 ls=80;

proc print data=mydata;
run;
```

SAS Log

The SAS log displays information about the successful import. For this example, PROC IMPORT generates a SAS DATA step, as shown in the partial log that follows.

```
 8    /*********************************************************************
 9    *   PRODUCT:   SAS
10    *   VERSION:   8.00
11    *   CREATOR:   External File Interface - Version 1.1
12    *   DATE:      01MAR1999
13    *   DESC:      Generated SAS Datastep Code
14    *   TEMPLATE SOURCE:  (None Specified.)
15    *********************************************************************/
16       data WORK.MYDATA                                    ;
17       %let _EFIERR_ = 0;     /* clear ERROR detection macro variable */
18       infile '/myfiles/mydata' delimiter = '&' MISSOVER
18 !  DSD lrecl=32767 firstobs=2 ;
19          format Region $8. ;
20          format State $2. ;
21          format Month $5. ;
22          format Expenses best12. ;
23          format Revenue best12. ;
24          informat Region $8. ;
25          informat State $2. ;
26          informat Month $5. ;
27          informat Expenses best32. ;
28          informat Revenue best32. ;
29       input
30                   Region $
31                   State $
32                   Month $
33                   Expenses
34                   Revenue
35       ;
36       If _ERROR_ then        /* ERROR detection */
37          call symput('_EFIERR_',1);
38       run;
NOTE: Numeric values have been converted to character
      values at the places given by: (Line):(Column).
      86:31
NOTE: 7 records were read from the infile
      '/myfiles/mydata'.
      The minimum record length was 27.
      The maximum record length was 28.
NOTE: The data set WORK.MYDATA has 7 observations and 5 variables.
NOTE: DATA statement used:
      real time           7.91 seconds
      cpu time            0.97 seconds



-42 rows created in WORK.MYDATA                                from
/myfiles/mydata.
NOTE: WORK.MYDATA was successfully created.
NOTE: PROCEDURE IMPORT used:
      real time           1:58.92
      cpu time            11.03 seconds
```

Output

This output lists the output data set, MYDATA, created by PROC IMPORT from the delimited external file.

```
                              The SAS System                                2

       Obs    Region      State    Month        Expenses         Revenue

         1    Southern      GA     JAN97            2000            8000
         2    Southern      GA     FEB97            1200            6000
         3    Southern      FL     FEB97            8500           11000
         4    Northern      NY     FEB97            3000            4000
         5    Northern      NY     MAR97            6000            5000
         6    Southern      FL     MAR97            9800           13500
         7    Northern      MA     MAR97            1500            1000
```

Example 2: Importing an Excel Spreadsheet

Procedure features:

Data source statements:

GETNAMES=
SHEET=

Other features:

PRINT procedure option:

OBS=

This example imports an Excel spreadsheet and creates a new, permanent SAS data set named SASUSER.ACCOUNTS.

Program

DATAFILE= specifies the input file. The filename contains the extension .XLS, which PROC IMPORT recognizes, for this example, as an Excel 5 spreadsheet.

```
proc import datafile="c:\myfiles\Accounts.xls"
```

OUT= identifies SASUSER.ACCOUNTS as the output SAS data set.

```
            out=sasuser.accounts
```

SHEET= specifies to import only the spreadsheet PRICES that is contained in the file ACCOUNTS.XLS.

```
    sheet="Prices";
```

GETNAMES= specifies that PROC IMPORT does not generate the variable names from the input file, but uses VAR0, VAR1, and so on.

```
   getnames=no;
run;
```

The PRINT procedure invoked with the OBS= option displays the first 10 observations of the output data set.

```
proc print data=sasuser.accounts(obs=10);
run;
```

Output

The following output displays the first 10 observations of the output data set, SASUSER.ACCOUNTS.

```
                                The SAS System                                1

OBS    VAR0                              VAR1                             VAR2

  1    Dharamsala Tea                    10 boxes x 20 bags              18.00
  2    Tibetan Barley Beer               24 - 12 oz bottles              19.00
  3    Licorice Syrup                    12 - 550 ml bottles             10.00
  4    Chef Anton's Cajun Seasoning      48 - 6 oz jars                  22.00
  5    Chef Anton's Gumbo Mix            36 boxes                        21.35
  6    Grandma's Boysenberry Spread      12 - 8 oz jars                  25.00
  7    Uncle Bob's Organic Dried Pears   12 - 1 lb pkgs.                 30.00
  8    Northwoods Cranberry Sauce        12 - 12 oz jars                 40.00
  9    Mishi Kobe Beef                   18 - 500 g pkgss.               97.00
 10    Fish Roe                          12 - 200 ml jars                31.00
```

Example 3: Importing a Microsoft Access Table

Procedure features:

PROC IMPORT statement arguments:

- TABLE=
- DBMS=

Data source Statements:

- DATABASE=
- PWD=
- UID=
- WGDB=

This example imports a Microsoft Access table and creates a permanent SAS data set named SASUSER.CUST. The Access table has user-level security, so it is necessary to specify values for the PWD=, UID=, and WGDB= statements.

Program

TABLE= specifies the input DBMS table name.

```
proc import table="customers"
```

OUT= identifies SASUSER.CUST as the output SAS data set.

```
            out=sasuser.cust
```

DBMS= specifies that the input file is a Microsoft Access table.

```
            dbms=access;
```

UID= identifies the user to the DBMS.

```
    uid="userid";
```

PWD= specifies the DBMS password used by the user to access the table.

```
    pwd="mypassword";
```

DATABASE= specifies the path and filename of the database that contains the table.

```
    database="c:\myfiles\east.mdb";
```

WGDB= specifies the workgroup (security) database name that contains the userid and password data for a Microsoft Access table.

```
    wgdb="c:\winnt\system32\security.mdb";
```

The PRINT procedure displays the first five observations of the output data set, SASUSER.CUST.

```
proc print data=sasuser.cust(obs=5);
run;
```

Output

The following output displays the first five observations of the output data set, SASUSER.CUST.

```
                                The SAS System                                1

Obs    Name                              Street                          Zipcode

  1    David Taylor                      124 Oxbow Street                72511
  2    Theo Barnes                       2412 McAllen Avenue             72513
  3    Lydia Stirog                      12550 Overton Place             72516
  4    Anton Niroles                     486 Gypsum Street               72511
  5    Cheryl Gaspar                     36 E. Broadway                  72515
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Procedures Guide, Version 8,* Cary, NC: SAS Institute Inc., 1999. 1729 pp.

SAS® Procedures Guide, Version 8

ISBN 1–58025–482–9

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

1st printing, October 1999