Paper 53

Advanced Methods to Introduce External Data into the SAS® System

Andrew T. Kuligowski, Nielsen Media Research

ABSTRACT / INTRODUCTION

The SAS® System has numerous capabilities to store, analyze, report, and present data. However, those features are useless unless that data are stored in or can be accessed by the SAS System. This presentation will provide a brief introduction to many of the different methods that can be used to pass data into the SAS System. Topics will include the menu-driven SAS Import Wizard, DDE, ODBC, and others - as many as time and space will allow!

The goal of this presentation is to provide information that will be useful to all users of the SAS System. Some topics are tailored to the novice, while others will be more applicable to the experienced user. Please note that some information, by its very nature, will only be applicable to select operating systems.

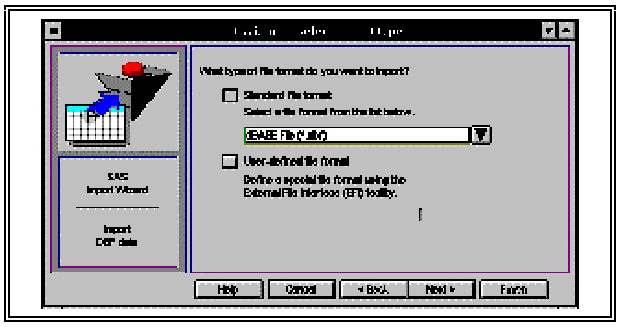
SAS IMPORT WIZARD

Often, the best tool for a job is the simplest one. To illustrate that point, the first topic we will discuss is the *SAS Import Wizard.* Available under Release 6.12 of the SAS System, the SAS Import Wizard is a menu-driven system to define and import external data into the SAS System. It is typical in appearance and in approach to Wizards available in other Windows products, such as Microsoft Excel. The Import Wizard is accessed by selecting **Import** from the **File** pulldown menu on the SAS toolbar.

The first screen requires the user to specify the type of file to be imported. The first choice is a "standard file format"; a pull-down menu allows various options such as dBASE, LOTUS, Excel, or delimited files. The other choice, "user-defined file format", provides an interface to the *External File Interface.* (The External File Interface, which permits the user to specify the details of an external file via menus, will not be discussed in this presentation.) **See Figure A for a sample of this screen.**

The SAS Import Wizard now displays a screen prompting the user to "Select File". The file name can be manually typed in the space provided, or the *Browse* button can be selected in order to search for the file. **See Figure B for a sample of this screen.**

Next, the user is prompted to "Select library and member". Again, the user can manually enter the SAS library and member into which the data is to be stored, or they can use pull-down menus to select from those which are already known to the SAS session. **See Figures C and D for samples of this screen.** If "standard file format" was originally selected, please note that the picture on the left side of the screen changes to a checkered flag when entry is finished, and the *Next* button becomes unusable, replaced by the *Finish* button. Otherwise, the checkered flag will be displayed on the next screen, which will advise the user that the Import Wizard is complete, and control will be transferred to the External File Interface.


What type of file format do you want to import?

☐ Standard file format

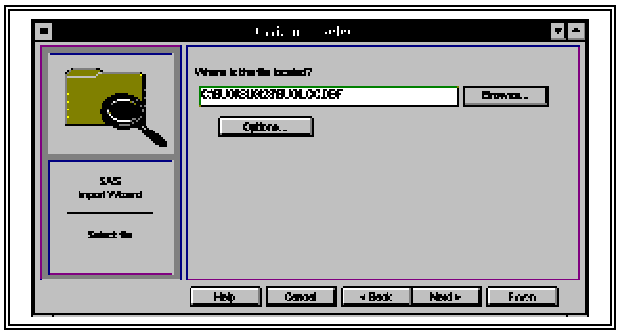
Select a file format from the list below.

dBASE File (*.dbf)

☐ User-defined file format

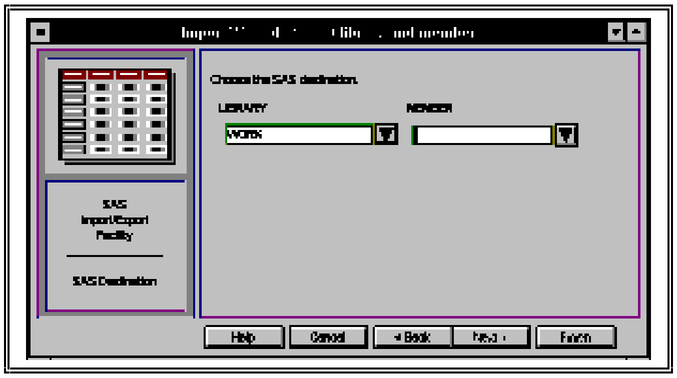
Define a special file format using the External File Interface (EFI) facility.

Figure "A" - SAS Import Wizard "Import Type" Screen


Where is the file located?

C:\BU08SUG23\BUDLOC.DBF

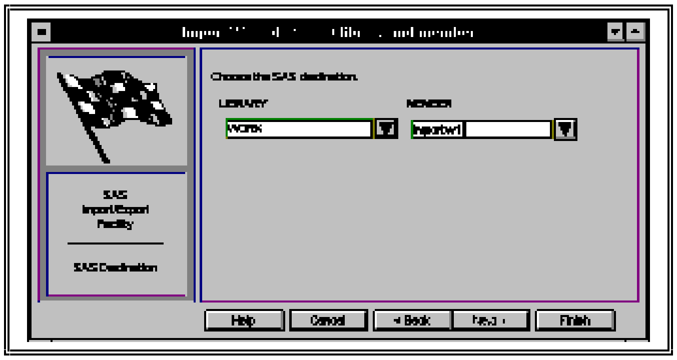
Figure "B" - SAS Import Wizard "Select File" Screen


Choose the SAS destination.

LIBRARY: WORK

MEMBER:

Figure "C" - SAS Import Wizard
"Library & Member" Screen (initial)


Choose the SAS destination.

LIBRARY: WORK

MEMBER: importwt

Figure "D" - SAS Import Wizard
"Library & Member" Screen (completed)

At this point, the SAS System processes the request.. Success is indicated by a simple message in the SASLOG, advising that the SAS dataset is now available for use :
NOTE: [*SASdsn*] was successfully created

The SAS Import Wizard is not always the optimal solution - if it was, this would be an awfully short presentation! There are two major drawbacks to the Import Wizard. First of all, it is not available when running SAS in an IBM mainframe environment. Secondly, it is an interactive tool, which renders it useless for a batch, schedule-oriented environment. However, the SAS Import Wizard can be the easiest, quickest, and therefore best solution for the one-time-only processing of an external file during an interactive SAS session.

CSV (Comma Separated Value) FILES

The *CSV,* or *Comma Separated Value* File is a special variety of sequential file, typically used for importing or exporting data from a spreadsheet. Data values are separated by commas, as is implied by the name, and character values are typically surrounded by double quotation marks (").

CSV files can be processed by using the DSD parameter (short for *D*elimiter-*S*eparated *D*ata) on the INFILE statement. This parameter automatically sets the default delimiter to comma, although this can be overridden by use of the DELIMITER= option. The presence of a pair of commas denotes a missing value. The DSD parameter also causes SAS to strip the double quotation marks, if present, from character values before storing them in SAS variables. Please note that character variables are defined with a default length of 8 bytes in this instance. This default length can be overridden by use of the **LENGTH** statement. Do not attempt to specify a format length on the input statement for character variables, as this may cause delimiting commas to be treated as part of the variable's value. **See Figure E for an example of reading a CSV file.**

```
DATA TEMP;
   LENGTH  CITY $ 20.  STATE $ 15. ;
   INFILE  CARDS  DSD ;
   INPUT  YEAR    CONFNAME $
          CITY $  STATE    $;
CARDS;
1998,” SUGI” ,” Nashville” ,” TN”
1998,” SESUG” ,” Norfolk” ,” VA”
1998,” WUSS” ,” Oakland” ,” CA”
1999,” SUGI” ,” Miami” ,” FL”
1999,” SESUG” ,” Mobile” ,” AL”
;
```

Figure "E" - CSV File

DDE

The next method that we shall discuss is *Dynamic Data Exchange*. Dynamic Data Exchange, or DDE, allows a client application to request information from a server application in a Windows or OS/2 environment. Effective with Release 6.08, the SAS System acts as a client application in this relationship. It can request data from a server application, with the requirement that the server application must be running. (It can also send commands and data to a server application, but that is a topic for another presentation.)

In order to use DDE, a connection must be established between the client application and the server application. This is accomplished by issuing a **FILENAME** statement with the keyword "DDE". The syntax for this statement, in this context, is:

    FILENAME *fileref* DDE *'DDE-triplet'* ;

The DDE-triplet is a specialized argument, and is made up of three components:

    application|topic!item

Application is the name of the server application, such as Excel. *Topic* is defined as the "topic of conversation"; basically, this is the file to be processed. *Item* is the "item of conversation"; in a spreadsheet, this is the range of cells that is to be included. For example, the DDE-triplet an Excel worksheet would be:

    Excel|[Book1]Sheet1!R1C1:R250C4

Note that the application and topic are separated by a vertical bar (|), while the topic and item are separated by an explanation point (!).

The DDE triplet for an application should be defined in the documentation for that application. However, most people find it easier to let SAS determine the proper DDE triplet. The following is a step-by-step method to obtain the proper DDE triplet for an application, assuming both SAS and that application are active:

- Toggle to your application, and use the standard PC "cut" techniques to store the portion of the client application to be processed in the Windows Clipboard. (For example, use the mouse to highlight the area to be "cut", then select CUT or COPY on the EDIT pop-up menu of most Windows applications.)
- Toggle to your SAS session, and click on the "Options" menu on the Menu Bar in SAS.
- The Options menu will contain an option called "DDE Triplet". Click on it.
- This will display an Information Box, which will contain the DDE-triplet.
- Enter this DDE-triplet into the FILENAME statement of your SAS routine.

If the user is willing to perform a little manual intervention, it is even possible to use DDE without ever knowing the name of the DDE triplet!

- As above, toggle to your application, and use the standard PC "cut" techniques to store the portion of the client application to be processed in the Windows Clipboard.
- Toggle to your SAS session, and replace the FILENAME statement with the following:

    FILENAME *fileref* DDE CLIPBOARD ;

The SAS routine is now ready to be executed. The weakness in this approach is that the data to be processed must be stored in the Clipboard prior to each invocation of your SAS routine. The benefit is that there is no need to ever know the DDE-triplet for your application to use DDE.

(Please note that this approach will only work if an application is DDE compliant.)

In order to use DDE with the SAS System, the server application must be running while SAS is running. If the server application is not active, then it can be invoked from within the SAS session with the "X" command. However, the SAS options XSYNC and XWAIT must be turned off before issuing this command, or control will not be returned to the SAS session until that external application is closed -- this defeats the purpose of a DDE link! (Of course, the user could also simply toggle over to the Windows Program Manager and manually invoke the application.)

The actual transfer of data from the external application to the SAS System is done via the combination of an **INFILE** and **INPUT** statement. The actual code to accomplish this task looks exactly like the code to read a sequential file into SAS. **See Figure F for an example of reading an Excel 5.0 spreadsheet via SAS.** For further examples covering a number of PC products, please refer to “Technical Support Document #325 - The SAS System and DDE”, which is available on the SAS Institute web site.

```
OPTIONS  NOXSYNC NOXWAIT;
X 'C:\EXCEL\SASCONF.XLS' ;
FILENAME SASCONF DDE
   'Excel|[Book1]Sheet1!R1C1:R250C4';
DATA CONFSCHD;
   INFILE SASCONF;
   INPUT DAY TIME TITLE AUTHOR;
RUN;
```

Figure "F" - INPUT from EXCEL using DDE

SAS/ACCESS ENGINES

SAS/ACCESS software is available for a variety of host systems, covering traditional mainframe, personal computer, and UNIX environments. It provides a method to view and transfer data from several common database management systems (DBMS) and a number of common PC file formats, into the SAS system.

The ACCESS procedure can create *descriptor* files that will provide information about the data stored in the DBMS table or PC file format, and use that information to create a SAS data file. In addition, use of an *interface view engine* will allow SAS to read data from the file formats directly into SAS routines. The interface view engine is used by the SAS SQL procedure to directly access external data bases without leaving the SAS session. However, the SQL statements used in the procedure are beyond the scope of this Tutorial.

There are two types of descriptor files created by the ACCESS procedure: an *access descriptor* and a *view descriptor*. Access descriptors provide information regarding the structure of the file to be accessed. This includes data types, table names, and column names, as well as the related SAS dataset information such as variable names and formats. This access descriptor can then be used to create the view descriptor which will contain criteria to be used to select columns and rows from the selected DBMS table or PC file. The data can be used directly from the view descriptor in the SAS routine, or it can be extracted from the DBMS or PC file into a SAS data file.

The type of DBMS or PC file to be used is specified in the PROC ACCESS statement in the form:

`PROC ACCESS DBMS=`*`filetype`*

Filetype can take on many different values. To cite just a few examples, the user can obtain data from DB2®, SYBASE®, and ORACLE® by selecting filetypes *DB2*, *SYBASE*, and *ORACLE*, respectively. In a Windows environment, *XLS*, and *WKn* (where *n* is a valid version number) will allow the transfer of data from Excel or Lotus® spreadsheets, while *DBF* and *DIF* are obviously the filetype for interfacing with .DBF and DIF formatted files, respectively.

The actual access or view descriptor is then created with the following syntax:

`CREATE libref.member-name.ACCESS` or

`CREATE libref.member-name.VIEW`

The PROC ACCESS and CREATE statements are followed by a statement that identifies the name of the DBMS, DBF, XLS or other file that will be accessed. In addition, there are other editing statements; these provide information about the structure of the DBMS or PC file being accessed, and select columns to be viewed. **See Figure G for an example of using PROC ACCESS create a view, with a subsequent use of its output.**

```
LIBNAME VWLIB 'c:\confdat\';
PROC ACCESS DBMS=xls;
  CREATE vwlib.states.access;
  PATH 'c:\confdat\state.xls';
     <editing statements omitted>
  CREATE vwlib.states2.view;
     <select, format, and
      subset statements omitted>
RUN;

DATA _NULL_;
  SET vwlib.states2;
  <statements omitted>
RUN;
```

Figure "G" - PROC ACCESS : Creation and use of a View

The SAS view descriptor can be used in any PROC or DATA step just like a SAS data set. It is also possible to use PROC ACCESS to create a SAS dataset from the view descriptor. This is accomplished by issuing PROC ACCESS with the VIEWDESC=*libref.view-descriptor* and OUT=*libref.sas-data-filename* options. **See Figure H for an example of using PROC ACCESS create a view, with a subsequent use of its output.**

```
PROC ACCESS VIEWDESC=vwlib.states2
            OUT=vwlib.stdata ;
RUN;

PROC PRINT DATA=vwlib.stdata;
RUN;
```

Figure "H" - PROC ACCESS : Creation of a SAS Dataset

It is not possible to fully cover PROC ACCESS in the limited space of this paper -- there are a number of manuals dedicated to the topic! For further information, including details of the DBLOAD procedure that will transfer data from SAS to the assorted DBMS and PC products, the reader is directed to the assorted SAS/ACCESS manuals.

PROC DIF and PROC DBF

SAS/ACCESS to PC File Formats, which requires a separate license from Base SAS software, also provides additional interfaces to two traditional types of PC files. *DBF* and *DIF* formats date back to the “ancient” days of PCs in the early 1980s; however, data continues to exist in these formats and one must be prepared to deal with it.

DBF files contain data originally formatted for the dBASE™ database management product, and are accessed through PROC DBF. PROC DBF accepts two options. The first, OUT= should be self-explanatory to anyone with even a modest experience in use of the SAS System. The other, DBn=, is required, and contains the file reference to the DBF file. The *n* in “DBn” refers to the version of dBASE for which the file was created; “2”, “3”, “4”, and “5” are valid version numbers. Please note that all dBASE formatted files are assigned the extension .DBF, regardless of the version of dBASE (or other product) under which they were created. SAS will produce an error message if the version number is incompatible with the file format.

PROC DBF will produce a SAS dataset, with variable names mapped from the original dBASE field names. Field names will be truncated to 8 characters, if necessary, to conform to current SAS variable name restrictions. **See Figure I for examples of PROC DBF.**

```
10   /* This is a dBASE IV format file. */
11   filename sasconf
        'c:\sasconf\confloc.dbf';
12   proc dbf db2=sasconf  out=conf_db2; RUN;

ERROR: Input file is not a DBASEII file.
WARNING: Data set WORK.CONF_DB2 not replaced
         because new file is incomplete.
NOTE: The SAS System stopped processing this
      step because of errors.
NOTE: The PROCEDURE DBF used 0.44 seconds.

13   proc dbf db4=sasconf  out=conf_db4; RUN;

NOTE: 7 observations written to the output
      SAS data set.
NOTE: The PROCEDURE DBF used 0.38 seconds.
```

Figure "I" - PROC DBF - Invalid and Correct

DIF files, short for Data Interchange Format, date back to VisiCalc™ and other early PC spreadsheet products. PROC DIF can be used to convert DIF files into SAS datasets. The format for PROC DIF is similar to PROC DBF, with the DIF= option used in place of DBn=. Variable names are assigned COL1 to COLn in the output SAS dataset; it is recommended that the user subsequently reassign them to something more meaningful. **See Figure J for an example of PROC DIF.**

```
filename sasconf 'c:\sasconf\confloc.dif';
proc dif dif=sasconf  out=conf_dif;
run;
proc print uniform;
run;
```

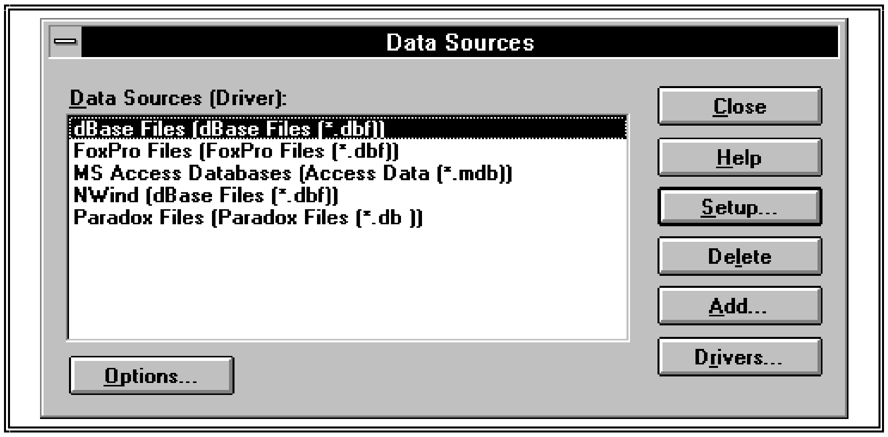
Figure "J" - PROC DIF

ODBC

Open DataBase Connectivity, or *ODBC*, started off as a standard for the exchange of data between DataBase Management Systems (DBMS) under Microsoft's Windows environment. Since those early days, interfaces to other operating systems and machines, such as the Apple MacIntosh, have been developed. It is necessary to use the SAS/ACCESS Interface to ODBC in order to use ODBC to bring data into the SAS System.

It is important to note that the SAS / ODBC interface does not directly obtain data from an external source, unlike other data sources which are available via SAS/ACCESS. Instead, it interfaces with the ODBC manager, which in turn interfaces with the other external data sources. In order to use the SAS / ODBC Interface, it is therefore first required to install the appropriate drivers, or “*Data Sources*”.. This is done via the ODBC Icon available through the operating system. (In both Windows 3.1 and Windows 95, for example, this can be found in the “Control Panel”.) **See Figures K, L, M, and N for examples of the screens used to define a SAS driver in ODBC.**

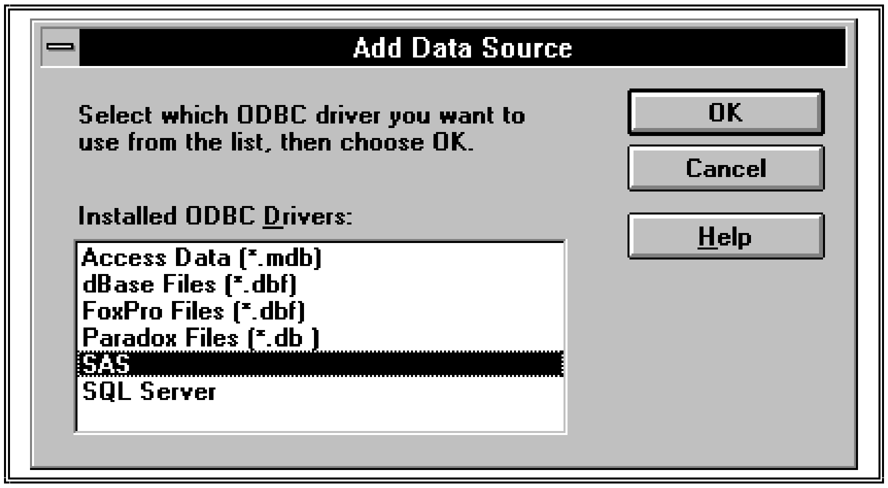
The SAS / ODBC interface is also unlike the other products in the SAS/ACCESS family, in that PROC ACCESS is not used to bring external data into SAS. Instead, the SAS System utilizes an enhancement to **PROC SQL**, the *SQL Procedure Pass-Through Facility*. (Please note that it is still necessary to license and install SAS/ACCESS to ODBC in order to use the SQL Pass-Through Facility in conjunction with ODBC. It should also be noted that the SQL Pass-Through Facility can also be used with other databases via separate SAS/ACCESS licenses; these will not be discussed in this presentation.)



Data Sources (Driver):

dBase Files (dBase Files (*.dbf))
FoxPro Files (FoxPro Files (*.dbf))
MS Access Databases (Access Data (*.mdb))
NWind (dBase Files (*.dbf))
Paradox Files (Paradox Files (*.db))

Figure "K" - Windows 3.1 ODBC Driver Configuration - Active Data Sources

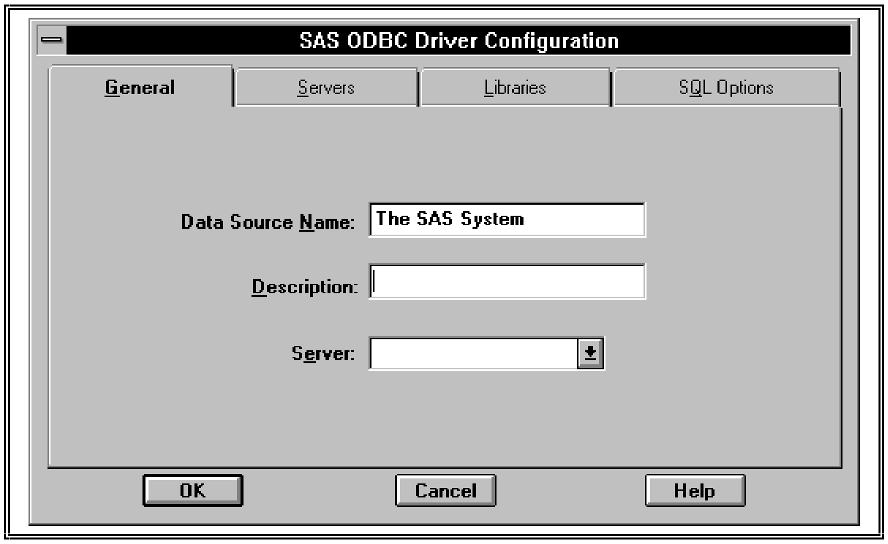


Select which ODBC driver you want to use from the list, then choose OK.

Installed ODBC Drivers:

Access Data (*.mdb)
dBase Files (*.dbf)
FoxPro Files (*.dbf)
Paradox Files (*.db)
SAS
SQL Server

Figure "L" - Windows 3.1 ODBC Driver Configuration - Add Data Source

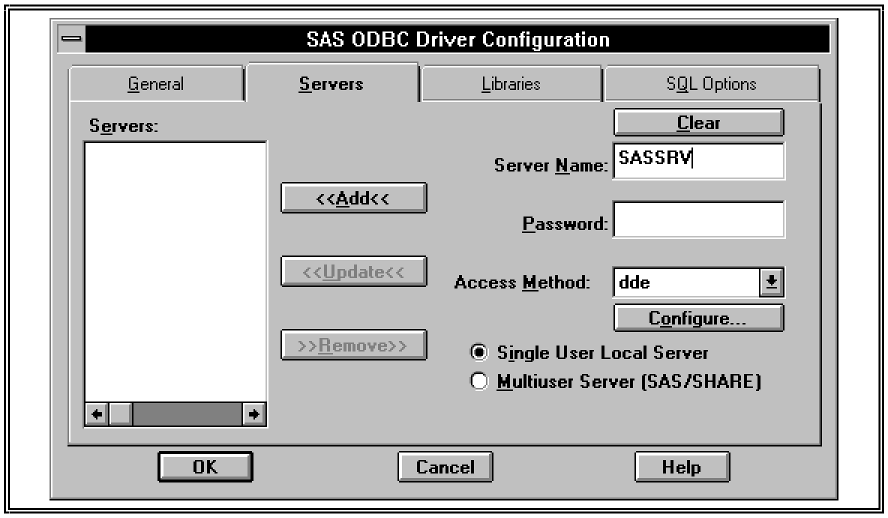


Data Source Name: The SAS System

Description:

Server:

Figure "M" - Windows 3.1 ODBC Driver Configuration - General Information



Server Name: SASSRV

Password:

Access Method: dde

[✓ Single User Local Server] Multiuser Server (SAS/SHARE)

Figure "N" - Windows 3.1 ODBC Driver Configuration - Servers

The SQL Procedure Pass-Through Facility has four statements associated with it; or more correctly, three statements and a "component", which is included within the SQL itself.

The first statement, **CONNECT**, establishes a connection with ODBC. The syntax is:

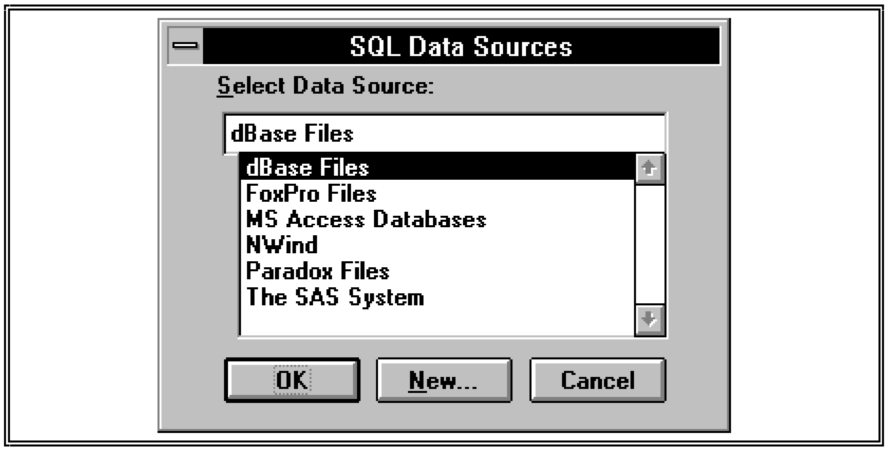
```
CONNECT TO ODBC <AS alias> <(options)>
```

The alias is optional; however, if used, the word "AS" *must* immediately precede it. The options are used to specify the data source which is to be processed, and must be enclosed within parentheses. **DSN=***data-source* can be used to specify a previously defined data source, as described above. (Note that "DSN" stands for "Data Source Name", and not "Data Set Name".) Alternatively, the word **PROMPT** will walk the user through the process of setting up an ODBC interface at execution time. Note that PROMPT and DSN= are mutually exclusive of each other. **See Figures O and P for an example of the screens displayed to define a dBASE data source.** **See Figures Q and R for the SAS source code to use these screens.** One other option that should be noted is **LOG**, which will cause all warnings and errors generated by the ODBC API (*Application Programming Interface)* to be written to the SASLOG. This may be especially useful during the development phase of an application.

As one would expect, **DISCONNECT** is the opposite of CONNECT. This statement ends the connection with ODBC. The syntax is :

```
    DISCONNECT FROM ODBC | alias;
```

using either the word "ODBC" or the alias that was specified in the CONNECT statement. It is optional, as an implicit DISCONNECT is issued when the **QUIT** statement ends PROC SQL. However, it may be desired to explicitly break the connection if more SQL statements are to be issued, or if a different CONNECT is desired.



dBase Files

dBase Files
FoxPro Files
MS Access Databases
NWind
Paradox Files
The SAS System

Figure "O" - SQL Pass-Through Facility CONNECT TO ... PROMPT / Data Source

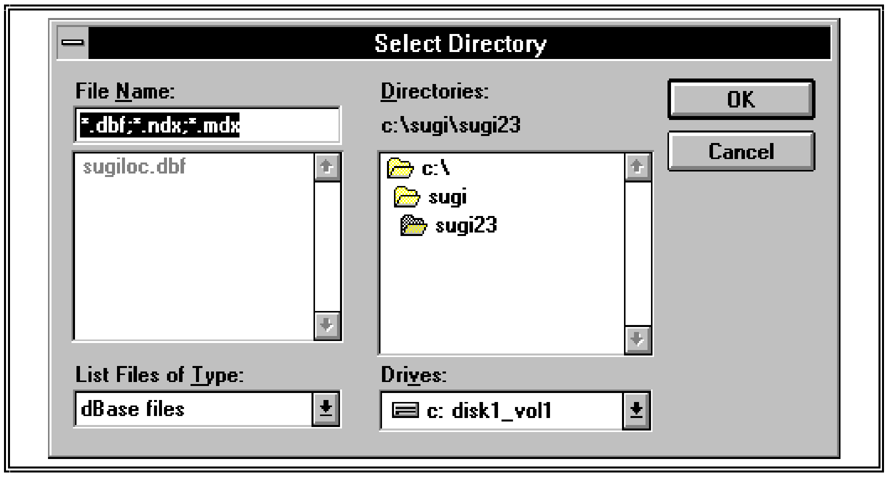


Figure "P" - SQL Pass-Through Facility CONNECT TO ... PROMPT / Select Directory

```
84   /* “ feedback” option expands SQL */
85   /* statements in the SASLOG.     */
86   proc sql feedback ;
87      connect to odbc(prompt);
88      create table confloc as
89         select * from connection to odbc
90           (select * from confloc );
NOTE: Statement transforms to:
        select YEAR, CONFEREN, CITY, STATE
          from connection to ODBC
                  /* dbms=ODBC,
                 connect options=(prompt) */
                  ( select * from confloc );
NOTE: Table WORK.CONFLOC created, with
      7 rows and 4 columns.
91      disconnect from odbc ;
92      quit ;
NOTE: The PROCEDURE SQL used 7.91 seconds.
```

Figure "Q" - SQL Pass-Through Facility using PROMPT

```
218  proc sql ;
219     connect to odbc as sasconf
220             (dsn="confdemo");
221     create table confloc as
222       select * from connection
                   to sasconf
223         (select * from confloc );
NOTE: Table WORK.CONFLOC created, with
      7 rows and 4 columns.
224     disconnect from sasconf ;
225     quit ;
NOTE: The PROCEDURE SQL used 3.35 seconds.
```

Figure "R" - SQL Pass-Through Facility using DSN=

The **EXECUTE** statement sends non-query SQL statements to ODBC. These are primarily used to update the external database, and as such are outside the scope of this presentation.

CONNECTION TO is a clause that can be inserted in an SQL **SELECT** statement to send an SQL query to the external data source accessed via ODBC. The syntax is:

```
    FROM CONNECTION TO ODBC | alias
          (SQL to external source)
```

again using either the word “ODBC” or the alias that was specified in the CONNECT statement. There are a number of special queries that can be used in conjunction with ODBC; these will be omitted from this presentation due to space limitations.

All of these statements and components can be combined with the statements normally associated with PROC SQL to form a successful ODBC query. **See Figure Q for an example of PROC SQL using PROMPT. See Figure R for an example of PROC SQL using DSN=.**

OLE

Object Linking and Embedding, or *OLE*, is another client/server methodology which will allow the transfer of data between SAS and other products. In the SAS System,® OLE functionality is typically attained through SAS/AF ® using FRAME catalog entries or through SAS/EIS ® , by invoking other products from within SAS. However, for our purposes, the SAS OLE Automation Server will allow other products to interface with Base SAS.

OLE is more graphically oriented than DDE; OLE will permit the SAS System to share a number of different objects, such as graphs and charts, with other products. As the name implies, OLE consists of an *object linking* process and an *object embedding* process. The object link permits an object to be updated on either the client or server side of the link, with the change reflected on the other side of the link. An embedded object, on the other hand, can only be changed from the client application.

The SAS OLE Automation Server is only supported for the Windows 95 and Windows NT operating systems.

THIRD PARTY PRODUCTS

In addition to the methods built into the Base SAS product and additional modules from SAS Institute, there are products produced by third-party providers that facilitate the transfer of data into SAS. In some cases, the user base for a given product does not warrant a separate SAS/ACCESS engine. The vendor for that product may provide their own interface to the SAS System. In other cases, an entrepreneur may believe that they have found a niche market which they believe SAS Institute has overlooked, or that they can provide an interface which is faster / quicker / cheaper / etc.; the marketplace will determine whether or not they have succeeded.

One example of a proprietary third-party product is PROC SAGE™. The Computer Corporation of America (CCA) supplies and supports a mainframe database called Model 204®. They also supply an interface called PROC SAGE, which passes data from a Model 204 database into a SAS dataset.

Conceptual Software, Inc. and Circle Systems both market products which they assert can be used to transfer external data into the SAS System under both Windows (3.1, '95, and NT) and various UNIX platforms. Conceptual Software, Inc. has two products that fit this category : DBMS/COPY™, to provide connectivity between over 80 different external data sources, including the SAS System; and DBMS/Engines™, to provide engines which allow the SAS System to interact with all of the products that can interface with DBMS/COPY. Circle Systems also has a product, Stat/Transfer™, which can be used to transfer data between SAS datasets and various other products. NOTE: It is not the purpose of this presentation or its author to either endorse or condemn any of these 3rd party products or their respective vendors. All claims are summarized based solely on information found on the companies' web sites.

There are also some utilities available in the public domain - the proper tool for your needs may already be available at no additional cost! The Proceedings from SUGI and from the various Regional SAS User Group conferences are good places to look, as are the SAS-L archives and the World Wide Web. For example, there are at least two published methods to bring the contents of a .dbf file into SAS; ironically, each is called DBFTOSAS. The first is a PERL script, written by Frank Stetzer, which will create two sequential files from a .dbf file - one containing the file's data in a sequential format, and the other being a SAS routine to process the first file. The other is a SAS macro, written by Richard Hockey. (The URLs for the Web sites containing these utilities can be found at the end of this presentation.)

ADDENDUM : VERSION 7 OF THE SAS SYSTEM

In November of 1998, SAS Institute began to ship the long-awaited Version 7 of the SAS System to a specially-selected group of SAS sites.. This final section of the document will discuss the changes and enhancements that Version 7 can now provide to the topics discussed in this presentation.

Most SAS routines written in Version 6 should be able to run under Version 7 without alterations. Therefore, the information contained in this presentation should not be rendered obsolete by the new release. However, the release will have a number of improvements, including some enhancements that should facilitate the transfer of external data into the SAS System. These include:

- **Variable names** will be increased to a maximum of 32 characters in length. These names will be stored in mixed case, although subsequent use of the variable name will not be case-sensitive. In addition, the SAS System can waive the normal variable name rules for SAS/ACCESS, including the allowance of blanks and special characters, through the use of the new SAS system option **VALIDVARNAME=ANY**. These features should combine to facilitate the use of field names from external DBMS - without having to convert them to an 8 character name for use within SAS.

- The **maximum length** of character fields has been increased from 200 to 32,767 characters.

- The **_INFILE_** pseudo-variable can now be used under the same circumstances as a regular variable. Formerly, it could only be used in conjunction with the PUT statement.

- **SCANOVER**, a new option on the INFILE statement, will permit the INPUT statement to automatically scan the next line when using @"textstring" construct, should the desired string not be found on the current input data line.

- **PROC IMPORT**, a new feature, will permit the use of the Import Facility within a SAS routine. Currently, this is only available via a series of interactive menus. In addition, the Import Facility will now support Excel 97 (*.xls) and Microsoft Access (*.mdb) database files.

- **SAS/ACCESS** definitions can now be made via a LIBNAME statement - ACCESS and VIEW tables are no longer required. This will allow the SAS System to refer to and use an external database as though it was a SAS data library. As a result, full-screen support for the ACCESS and VIEW descriptors will no longer be provided.

- SAS/ACCESS engines for **AS/400** and **MS SQL** have been dropped from Version 7. Users are encouraged to move to the SAS/ODBC interface - a new **PROC CV2ODBC** has been provided to facilitate the conversion. The ODBC Engine itself is also easier to use under the new release.

CONCLUSION

There are a number of methods to introduce external data into the SAS System. It would be impossible to provide in-depth information on all of them in the limited space of this presentation. It is hoped that the material contained in this paper will serve to stimulate the curiosity of the reader, and that they will continue their education by researching the appropriate manuals and technical papers devoted to the specific topics discussed within this paper. Ultimately, however, it will be through real-life trial and error that true comprehension and retention of this knowledge will be attained.

REFERENCES / FOR FURTHER INFORMATION

Beatrous, Steve, and Clifford, Billy. (1998). “Sometimes You Get What You Want: I/O Enhancements for Version 7”. *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Bodt, Mark (1996). “Talking to PC Applications Using Dynamic Data Exchange”. *Observations*, Volume 5, No. 3 (Second Quarter 1996). Cary, NC: SAS Institute, Inc.

Boling, John C. (1997). “SAS Data Views: A Virtual View of Data”. *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Carey, Helen and Carey, Ginger (1996). *SAS Today! A Year of Terrific Tips.* Cary, NC: SAS Institute, Inc.

Clegg, Jennifer, and Rigsbee, Carol (1997). “OLE and the SAS System for Windows Release 6.12”. *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Cody, Ronald (1998). “The INPUT Statement: Where It’s @”. *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Dickson, Alan, and Pass, Ray (1996). “SELECT ITEMS FROM PROC.SQL Where ITEMS > BASICS”. *Proceedings of the Twenty-First Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Gilmore, Jodie (1997). “Using Dynamic Data Exchange with Microsoft Word”. *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Gona, Vino, and Van Wyk, Jana (1998). “Version 7 Enhancements to SAS/ACCESS Software”. *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Heffner, William F. (1998). “DATA Step in Version 7: What’s New?”. *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Kuligowski, Andrew T., and Roberts, Nancy (1997). “From There to Here: Getting Your Data Into the SAS System”. *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Kuligowski, Andrew T. (1998). “An Overview of Techniques to Introduce External Data into the SAS System”. *Proceedings of the Sixth Annual Conference of the SouthEast SAS Users Group*. USA.

Levine, Allison. (1997). “The What, When, Why, and How of PROC FORMAT”. *Proceedings of the Tenth Annual NorthEast SAS Users Group Conference*. USA.

Ma, Meimei, and Schlotzhauer, Sandra (1998). “Tips and Techniques for Moving Between Operating Environments”. *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Mason, Phil. (1996). *In the Know … SAS Tips and Techniques from Around the Globe..* Cary, NC: SAS Institute, Inc.

Praxis International, Inc. (1993), *SAGE/204 User's Guide Release 2.1*. USA: Praxis International, Inc.

Riba, S. David (1996), *Course Notes: Connecting With Your Data*. Clearwater, FL: JADE Tech, Inc.

Riba, S. David, and Riba, Elisabeth A. (1996), "ODBC: Windows to the Outside World". *Proceedings of the Fourth Annual Conference of the SouthEast SAS Users Group*. USA.

Riba, S. David (1996), "Open DataBase Connectivity and SAS". *The NESUG Express*, October 1996 Edition.

Sanders, Roger E. (1998). “Accessing Data from Your PC Using Version 7 of the SAS System”. *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1995), *DBF and DIF Procedures, Release 6.11, Preliminary Documentation.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1993), *SAS Companion for the Microsoft Windows Environment.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1994), *Getting Started with SAS/ACCESS Software, Version 6, First Edition.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1994), *SAS/ACCESS Software Changes and Enhancements : SQL Pass-Through Facility, Version 6.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1995), *SAS/ACCESS Software for PC File Formats: Reference, Version 6, First Edition.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1994), *SAS/ACCESS Software for Relational Databases: Reference, Version 6, First Edition.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1990), *SAS Language: Reference, Version 6, First Edition.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1994), *SAS ODBC Driver Technical Report: User's Guide and Programmer's Reference, Version 6.10.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1996). *SAS Online Documentation.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1990). *SAS Procedures Guide, Version 6, Third Edition.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1994), *SAS Software: Abridged Reference, Version 6, First Edition.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1989). *SAS Technical Report P-195, Transporting SAS Files between Host Systems.* Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1991). *SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07.* Cary, NC: SAS Institute, Inc.

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

Web Sites

Circle Systems, "Stat/Transfer. Home Page". *http://www.stattransfer.com.*

Conceptual Software, Inc., "Conceptual Software, Inc. Home Page". *http://www.conceptual.com.*

Cram, Donald P., "Excel 2 SAS and Back Webpage". *http://www-leland.stanford.edu/class/gsb/excel2sas.html.*

Hockey, Richard, "DBFTOSAS Macro". *http://ftp.urz.uni-heidelberg.de/ftp/put/sas/makros/dbtosas.sas..*

Riba, S. David, "JADE Tech Publications". *http://www.jadetek.com./publicat.htm.*

SAS Institute, Inc., "Technical Support Document #325 - The SAS System and DDE". *http://www.sas.com/techsup/download/technote/TS325.doc.*

SAS Institute, Inc., "Technical Support Document #501 - Data Acquisition and Exportation in PC SAS". *http://www.sas.com/techsup/download/technote/TS501.pdf.*

Stetzer, John, "DBFTOSAS PERL Script". *http://www.iiug.org/memb_software/archive/dbftosas.perl.*

The author can be contacted via e-mail as follows:
Andrew T. Kuligowski
0005949476@mcimail.com
kuligoat@tvratings.com

ACKNOWLEDGMENTS

I would like to acknowledge and thank Ms. Nancy Roberts. Nancy was the co-author of the ancestor of this paper, *From There to Here: Getting Your Data Into the SAS® System*, which can be found in the SUGI 22 Proceedings. Some portions of this presentation can be traced back to Nancy's work, and are used with her permission. I would also like to thank Nancy, along with Linda Kuligowski, and S. David Riba for their assistance and support during the production of this document.