

Data Acquisition and Exportation in PC SAS

There are many different types of data that can be read into the SAS System. The same is true for exporting SAS datasets out of the System. Sometimes you may have the option to choose between several different methods for reading/writing your external data. Each method has its own limitations, advantages, and disadvantages. In this document, we will discuss each method, the types of data that it can access/write, and what limitations, advantages, and disadvantages apply to that method. We will be concentrating on methods available in the PC (WINDOWS and OS/2) environments, although many of these topics are portable to other operating systems also.

ASCII File (Base SAS Software - all versions and platforms)

The simplest way to read or write data is through an ASCII file. The only tool needed for this process is Base SAS Software. An ASCII file is a file that is made of plain ASCII text, there are no special characters and no encryption. These types of files are easily read into text editors like Notepad or E, and most spreadsheet and database products. You have all the flexibility you need to layout or read the file according to the guidelines of the other product. For instance, you may need to specify certain delimiters to use between each column of data, specific formats for data, or column positions. Here is an example of creating an ASCII file from a dataset (sasuser.houses), using tab delimiters (tabs are a hex '09' character), no fixed column positions:

```
DATA _NULL_;
FILE 'C:\MYFILE.TXT';
SET SASUSER.HOUSES;
PUT STREET '09'x STYLE '09'x PRICE DOLLAR12.;
RUN;
```

If a file of this same structure were created by another software package, we could read it in and create a SAS dataset with this dataset:

```
DATA SASUSER.NEW;
INFILE 'C:\MYFILE.TXT' DLM='09'x TRUNCOVER LRECL=132 ;
LENGTH STREET $ 16;
INPUT STREET $ STYLE $ PRICE DOLLAR12.;
RUN;
```

A third example below writes the widely-used “comma separated file” with as little coding as possible. This code does not require typing in the list of variable names on the PUT statement, which can be very time-consuming, but it does force the variable ordering to be grouped by variable type. For instance, in this example, we will have an external file with all of our character variables at the beginning of the record in the order that they appear in the SAS dataset, followed by all of the numeric variables.

```
DATA A;
INPUT A $ B C $ D;
CARDS;
W 1 X 2
Y 3 Z 4
DATA _NULL_;
SET A;                               /*set your sas data set*/
FILE 'EXTERNAL-FILE';                /*name your external file here*/
ARRAY CHAR (*) $ _CHARACTER_;       /*assigns all your character vars to this array*/
ARRAY NUM (*) _NUMERIC_;            /*assigns all your numeric vars to this array*/
PUT (CHAR(*) ($ +(-1) ',')) @;
PUT (NUM(*) ( +(-1) ','));
RUN;
```

This code would result in the following two lines being written to the external file:

```
W,X,1,2
Y,Z,3,4
```

Reading “comma separated files” is handled well in SAS using the DSD option on the INFILE statement. See *Technical Report P-222* for full documentation on the option.

Further documentation:

“SAS Language, Reference, Version 6, First Edition” Chapter 2 The Data Step

“SAS Language, Reference, Version 6, First Edition” Chapter 9 SAS Language Statements

“Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07” pp22-46

Dynamic Data Exchange - DDE (Base SAS Software - all versions of Windows & OS/2 SAS)

DDE is a feature of Base SAS Software that allows you to exchange data back and forth between other open packages such as word processors, spreadsheets, or database packages that are also DDE compliant. SAS acts as a client in this process which means it can send data, send commands, and retrieve data. Other packages can not send commands or retrieve data from SAS, as that is the function of a server application. This method is very similar to reading and writing ASCII files, except that the file referenced is a special path that physically points to an open application on your PC. The best and easiest way to determine this special file reference is by following the steps in the section of your SAS Companion entitled ‘Determining the DDE Triplet’. Once you have setup the proper file reference to tell SAS where your data resides or where you would like to write your existing SAS dataset, the rest of the program follows the guidelines above using the INPUT or PUT statements to lay out the record format.

Further documentation:

“SAS Companion for the OS/2 Environment, Version 6, Second ed.” pp124-132

“SAS Companion for the Microsoft Windows Environment, Version 6, First ed.” pp126-134

“SAS Companion for the Microsoft Windows NT Environment, Version 6, First ed.” pp132-140

Technical Support Document TS-325

SAS Engines (Base SAS Software)

SAS provides engines that are used on the LIBNAME statement in order to share data between non-SAS applications and different versions of SAS available on your platform.

For example, the engines V603, V604, V606, and V608 are designed to help you exchange datasets between the different versions of SAS that have been developed on the PC platforms. These engines are placed on a LIBNAME statement to allocate what type of SAS dataset SAS should look for or write when referencing the specified directory. For example, if we are in version 6.08 SAS for Windows, and we want to view the contents of C:\DOG.SSD, a SAS dataset from version 6.04 SAS for DOS, we would use statements similar to:

```
LIBNAME MYDATA V604 'C:\';  
PROC PRINT DATA=MYDATA.DOG; RUN;
```

In whatever context you reference the libref MYDATA, SAS will know to use version 6.04 standards when reading and writing. Each version of SAS will have engines that reference the prior versions of SAS (i.e., version 6.04 SAS does not have a V608 engine, but version 6.08 SAS does have a V604 engine). This example converts version 6.08 datasets to 6.11:

```
LIBNAME OLDDATA V608 'C:\608DATA\';  
LIBNAME NEWDATA V611 'C:\611DATA\';
```

```
PROC COPY IN=OLDDATA OUT=NEWDATA; RUN;
```

or

```
DATA NEWDATA.TEST;  
SET OLDDATA.TEST; RUN;
```

Another set of engines available in releases 6.06 and above of the Windows and OS/2 SAS systems are the interface engines. The three interface engines in the PC environment allow read-only access to the file formats of BMDP, SPSS, and OSIRIS. These engines are also used on the LIBNAME statement, but the path that the libref points to must be the name of the physical file. For example:

```
LIBNAME MYDATA SPSS 'C:\ASPSS.DAT';
```

```
PROC PRINT DATA=SPSS._FIRST_; RUN;
```

or

```
DATA NEW; SET SPSS._FIRST_; RUN;
```

Further documentation:

“SAS Companion for the OS/2 Environment, Version 6, Second ed.” pp83-86

“SAS Companion for the Microsoft Windows Environment, Version 6, First ed.” pp84-87

“SAS Companion for the Microsoft Windows NT Environment, Version 6, First ed.” pp91-94

SAS/Connect

Another part of the SAS System that can help in data acquisition and exportation is SAS/Connect. SAS/Connect software enables a SAS Session on one host to create a SAS session on another host and use that remote session to access remote data and execute SAS statements. SAS/CONNECT software provides three types of services:

- o Remote compute services allow you to submit SAS statements to the remote session for execution and get the results back in the local session. Remote compute services give you easy access to the remote resources on your network.
- o Remote library services (RLS) allow you to access SAS data on a remote host regardless of machine architecture as if it were local. RLS is a feature of SAS/Connect and SAS/Share software. With this feature, you can submit SAS statements to the local machine processor and have it access data that reside on the remote machine.
- o Data transfer services allow you to explicitly upload and download files from one host to the other, including SAS data files and catalogs, external data files, and binary files.

Along with the straight forward accesses described above, consider the task of creating a Windows SAS dataset from a mainframe DB2 table. Windows SAS does not have an Access product to DB2, but MVS SAS does; so one could use SAS/Connect to link Windows SAS to MVS SAS, then remote submit Proc ACCESS code against SAS/Access to DB2 that resides on the MVS host, and finally download the resultant table or view. Basically, SAS/Connect is a tool that multiplies the access methods of two different platforms of SAS together to form many new combinations.

Further documentation:

“SAS/CONNECT Software, Usage and Reference, Version 6, Second edition”

Proc COPY: SAS Transport files (Base SAS - all platforms and versions)

These procedures are used to create SAS Transport files from native SAS datasets. After a SAS file has been converted into a transport file, it can be moved to any other platform that has SAS. Transport files can contain one or more datasets. On platforms where files must be allocated before being written, a transport file must be created with the structure of LRECL=80, BLKSIZE=8000, RECFM=FB, as illustrated below .

!Attention JMP users!

Transport files are also the best way to exchange data between SAS and the JMP software products. In JMP, both the IMPORT and SAVE AS dialogs refer to SAS Transport File as a valid file type to read in or write out.

The following steps illustrate how to move a SAS dataset from MVS to PC SAS using a transport file:

1. This code should be run on the MVS SAS system:
TSO ALLOC FI(TRANS) DA('SASPLV.TRANS.DAT') NEW DSORG(PS)
RECFM(F,B) LRECL(80) BLKSIZE(8000);
LIBNAME TRANS XPORT;
LIBNAME INDATA 'SASPLV.SAS.DATA';
PROC COPY IN=INDATA OUT=TRANS; RUN;
2. Move the transport file from MVS to the PC with any file transfer software, ensuring that a BINARY transfer is specified (i.e., no EBCDIC to ASCII translation).
3. Read in the transport file with this code on the PC:
LIBNAME TRANS XPORT 'C:\TRANS.DAT';
LIBNAME OUTDATA 'C:\MYDATA\';
PROC COPY IN=TRANS OUT=OUTDATA; RUN;

If the operating system from which you are exporting does not require file allocation, step 1 would begin with the line:

```
LIBNAME TRANS XPORT 'external-file';
```

where 'external file' is the full path, including filename, to the transport file you are attempting to create.

Further documentation:

“SAS Procedures Guide, Version 6, Third edition” pp204-208

“SAS Technical Report P-195, Transporting SAS Files between Host Systems”.

ACCESS PRODUCTS : General Information

Before we start discussing SAS/Access products and code in any detail, let's go over some of the components that will be referenced in the portions of the document to follow....

INPUT SAS DATASET= the dataset that currently exists in SAS (permanent or temporary) that we would like to export out to some other file type.

OUTPUT SAS DATASET= the SAS dataset that we would like to create by importing some other type of data that currently resides in an external file format.

EXTERNAL FILE= this refers to any type of data that is not native to SAS (i.e., not a SAS dataset).

ACCESS DESCRIPTOR= a SAS structure that points to an external file and contains the structural information about that file such that SAS is able to interpret the foreign data format.

ACCESS VIEW= a SAS structure that points to an Access Descriptor, gaining access to a piece of external data. This structure is used in the same context that a SAS dataset is used, although, whenever this structure is referenced, the data is extracted at that time from the external data source that the related Access Descriptor defines.

An Access Descriptor is a file type that references a specific external file type, defining its location and internal structure. A View Descriptor is a file type that uses an Access Descriptor to locate the file and determines what parts of the data to import to SAS. These View Descriptors are used in much of the same ways that SAS datasets are used. You can edit them with SAS/FSP, run diagnostic procedures against them, and set them in datastep code. As you modify an Access View, you are also modifying the external file that it references. Also, anytime the data of that external file is changed, and you subsequently reference the Access View to that file, the View will reflect the new data in the external file automatically.

Below is the list of Access products currently offered in the PC environments:

AS/400 :	OS/2
DB2/2:	OS/2
ODBC *:	WINDOWS, WINDOWS 95
ORACLE:	WINDOWS NT (6.11 TS025), OS/2
PC File Formats:	OS/2, WINDOWS, WINDOWS NT, WINDOWS 95
6.08 =	.DBF (versions 3&4), .DIF
6.10 =	.DBF (versions 3&4), .DIF, .WK1, .WK3
6.11 =	.DBF (versions 3-5), .DIF, .WK1, .WK3, .WK4, .XLS
SYBASE &	
SQL Server:	OS/2, WINDOWS, WINDOWS NT, WINDOWS 95

* PASS-THROUGH facility via PROC SQL only; PROCs ACCESS and DBLOAD are not supplied

PROC ACCESS non-interactive (SAS/Access)

Procedure Access is the tool provided in SAS/Access for creating access descriptors, access views, and subsequently SAS datasets from these descriptors and views. This procedure has full syntax for coding these actions for batch

processing. Following is an example for reading in a .DBF file:

```
PROC ACCESS DBMS=DBF;
CREATE SASUSER.CUSTOMER.ACCESS;
PATH='C:\DATA\CUSTOMER.DBF';
CREATE SASUSER.CUSTS.VIEW;
SELECT ALL; RUN;
PROC PRINT DATA=CUSTS; RUN;
```

PRINT ⇒reads⇒ CUSTS.VIEW ⇒reads⇒ CUSTOMER.ACCESS ⇒reads⇒ C:\DATA\CUSTOMER.DBF

This is the most simplified version of PROC ACCESS and although it refers to DBF, it is valid for all file types for which there is an Access product interface. In this example, the CUSTS view is referred to in the PROC PRINT

code in the same manner as we would refer to a SAS dataset. As the view is referenced though, the data is read from the external file before it is printed to the OUTPUT window.

If you no longer need the connection to the external file, you can create a SAS dataset from your view that will be standalone data, meaning that it will not rely on the external file for processing.

```
PROC ACCESS VIEWDESC=SASUSER.CUSTS OUT=SASUSER.CLIENTS;  
RUN;
```

Notice that the OUT= dataset has a different name than the view. Because datasets and views are referenced in the same manner, you cannot have a view and dataset of the same name.

There are many other options and statements you can use to enhance the functionality of the ACCESS procedure; see the full line of SAS/Access product documentation for the databases you have licensed.

PROC DBLOAD (SAS/Access)

Procedure DBLOAD is the counterpart to PROC ACCESS in that its goal is to write SAS Datasets out to some other specified file type. Note: If the file referenced in the PATH= options already exists, PROC DBLOAD will fail; it will not overwrite an existing file.

```
PROC DBLOAD DBMS=DBASE DATA=CUSTOMER;  
  PATH='C:\DATA\CUSTOMER.DBF';  
  LOAD;  
  RUN;
```

As mentioned in the section for Proc ACCESS, this is only an example of the functionality of Proc DBLOAD, for further details on optional statements, see the documentation for your ACCESS product.

PROC ACCESS interactive (SAS/Access)

PROC ACCESS can also be invoked interactively to allow you to work through interactive windows where you enter all the information needed to point SAS to your external file. No coding is involved with this method. You can type **ACCESS** at the SAS command line to invoke the application or click on **Globals... Access** from the menu bars or pop-up menus.

Further Documentation:

The SAS/Access documents have appendices that illustrate each window of the interactive version of Proc Access and Proc Dblload.

Query window (SAS/Assist all platforms version 6.08, 6.10, BASE SAS all platforms 6.11)

The Query window is an interactive application that creates SQL queries against all types of data. From this facility you can create views and tables, and write reports, using local datasets, remote datasets, and data being retrieved via a SAS/Access product. To activate a SAS/Access product from the Query window, select **Actions... Switch Access Mode**. Remember though, this interface relies on an underlying SAS/Access product if you are trying to communicate with external data types. You must have that SAS/Access product installed and licensed before you can use the Query window as the tool to drive that particular product. It is a read only facility and does not provide any means of outputting SAS data to any other external file type. It will interface to local datasets, SAS/Connect, and all SAS/Access products.

PROC DBF and DIF (SAS/Access to PC File Formats - Windows, Win NT, and OS/2 platforms)

When SAS/Access to PC File Formats is licensed with version 6.08 ts407 and above, included are the procedures DBF and DIF. These procedures were part of Base SAS under 6.04 for DOS and many of our users became quite accustomed to them. They are very simple procedures used for reading and writing the universal file formats of .DBF and .DIF, but they are also very limited in their capabilities. These procedures provide a quick, static copy of the current SAS input dataset or of the current input .DBF/.DIF file. Each time one of the files change, the conversion will need to be done again to reflect those changes. All subsetting and formatting must take place prior to exporting the SAS dataset or after importing the .DBF/.DIF file.

The example below writes the SASUSER.HOUSES dataset out to a .DBF file and then reads that .DBF file back in to create a new dataset named WORK.NEW:

```
FILENAME DBFFILE 'C:\DATA\HOUSES.DBF'; /* This file does not need to exist before
                                        exporting a SAS dataset to it */
PROC DBF DB4=DBFFILE DATA=SASUSER.HOUSES;
FORMAT PRICE DOLLAR11.2; RUN; /*sas -to-> dbf */
PROC DBF DB4=DBFFILE OUT=WORK.NEW; RUN; /* dbf -to-> sas */
```

Special considerations:

Proc DBF and DIF cannot convert files with more than 32767 observations. The DB4= option of Proc DBF cannot convert more than 255 variables. DBF MEMO fields will be ignored during input. Format statements must be used on all numeric variables with decimal data in Proc DBF.

Further documentation:

“SAS Procedures Guide, Release 6.03 Edition” Chapters 16 & 17
Technical Support Document TS-361

PROC SQL pass-through (SAS/Access to ODBC - all versions of Windows SAS)

SAS/Access to ODBC is not a full Access product; it is called a pass-through facility. That means that there is no Proc ACCESS and DBLOAD with this product. The functionality for getting to and from ODBC data sources is added to the BASE procedure of Proc SQL when this product is installed. With this product, you have the tool needed to establish a conversation to external data via the ODBC driver for that particular package. **Before attempting this type of connection, you will need to determine how to obtain, install, and setup the ODBC driver for the database to which you would like to gain access!** For example, we provide the SAS ODBC driver with BASE SAS software, which allows other applications to gain access to our data (see further discussion in last section of this document).

Once you have setup a DataSourceName (DSN) through ODBC manager and your installed ODBC drivers, you can then begin to use Proc SQL in a manner similar to this example:

```
PROC SQL;
CONNECT TO ODBC (DSN=MSACCESS);
CREATE VIEW SASUSER.TEST AS SELECT *
FROM CONNECTION TO ODBC (SELECT * FROM CUSTOMERS);
DISCONNECT FROM ODBC;
QUIT;
```

Explanation of integral parts of this example:

MSACCESS = the data source name that was defined after an ODBC driver was installed for the MS ACCESS database product, obtained from Microsoft. During setup of this data source name, the driver is selected and then you must setup pertinent information for the driver such as where your MS ACCESS database resides.

VIEW = see definition for access view discussed on page 4. One exception being that views created with PROC SQL cannot be modified or updated.

TEST = arbitrary name for the access view being stored in the SASUSER library

* = all variables available in the database

CUSTOMERS = the name of the specific data table. This data table is one of the components of the database that was setup during the creation of the data source name.

The CREATE VIEW statement could have also been entered as CREATE TABLE to create a SAS dataset during this procedure instead of an SQL view.

Hint: If you have trouble determining the DSN you previously set up, try substituting the word 'prompt' for the DSN= option used above. Like this:

```
PROC SQL;
CONNECT TO ODBC (PROMPT);
etc.....
```

One of the disadvantages to the ODBC pass-through facility is that it does not lend itself easily to writing SAS data out to another file type. This task is accomplished through the EXECUTE INSERT statement of Proc SQL, and there must be a separate statement for each observation added.

For example:

```
PROC SQL;
CONNECT TO ODBC (DSN=MSACCESS);
EXECUTE (INSERT INTO CUSTOMERS
VALUES('SAS', 'SAS', 'JOHN DOE', 'TECH SUPPORT', 'SAS CAMPUS DR.',
'CARY', NULL, '27513', 'USA', '677-8008', '677-4444')) BY ODBC;
%PUT &SQLXMSG;
DISCONNECT FROM ODBC;
QUIT;
```

This code would add one record to the CUSTOMERS data table.

For further examples, options, and discussion on connecting to ODBC data, see:

“SAS Technical Report P-262 SAS/Access Interface to ODBC: SQL Procedure Pass-Through Facility”.

For further capabilities of Proc SQL, see:

“SAS Guide to the SQL Procedure”.

The SAS ODBC Driver

The SAS ODBC driver is a tool that creates an ODBC connection from any Windows ODBC compliant application to SAS datasets. It is designed such that you can work in a Windows application (with ODBC capabilities) and import SAS datasets directly into the native format for that application. The steps required to make the connection are dependent on the application that you are using, but generally an IMPORT option or menu item will refer to SQL or ODBC data, that will then display the DSN's you have defined in ODBC Manager.

Our driver has two different access methods for connecting to data, TCP and DDE, found in the SERVER setup of the SAS ODBC driver setup. DDE would be used for connecting to a local (WINDOWS) SAS session and TCP would be used to connect to a remote operating system and access SAS remotely, such as UNIX or MVS.

In the case of a TCP connection, not only do you need BASE SAS installed on the remote operating system, you also need SAS/SHARE and SAS/SHARE*NET. Contact your SAS sales representative for pricing information on these two products.

Further documentation:

“SAS ODBC Driver Technical Report: User's Guide and Programmer's Reference”