ABSTRACT

In order to support the statistical analysis of clinical data, we utilized the DDE facility in SAS to retrieve clinical study data from a Microsoft Access database. Although the DDE facility is documented, it took quite a bit of experimentation to successfully execute. Our programming environment included Windows for Workgroups 3.11, SAS 6.10, and MS Access 2.0. Our Microsoft Access database was available to many users on a local area network and DDE provided us with the most current data. In our paper, we use our programs to provide examples to review the DDE syntax using the filename statement and explain how the DDE triplet maps to our MS Access database. Our example reads in entire Access tables, but we describe how to set limits. This paper also touches on accessing MSEexcel data from within SAS as well as writing data to Microsoft applications from within SAS. Our examples use SAS Macro code so accessing current data from the database could be easily repeated in any calling program. Our paper highlights the fine points and pitfalls of this useful DDE technique.

INTRODUCTION

We needed to analyze clinical study data from a Microsoft Access database using SAS programming tools. We considered four possible solutions: 1) exporting ASCII files from the Access database and writing an infile statement in the SAS System, 2) purchasing third party software that converts common PC databases into SAS datasets, 3) obtaining a license for SAS/Access®, and 4) using Dynamic Data Exchange (DDE) to exchange information between Windows applications. While the first option is a proven reliable system, it involves several steps, programming and extra storage. In addition it is quite time consuming and the result is a static copy of your data. The second option is simply a variation of the first and has the same drawbacks except for programming. It is also expensive and the third party software versions must be kept up to date. The third option was not feasible due to timing and cost concerns. We decided on the final solution, DDE, as the best alternative since it promised direct access to the data. Current data could be read from the database without extra storage requirements. It did require some programming, but it was very efficient and reading the data was accomplished in one step.

TECHNICAL DISCUSSION

We reviewed documentation in the SAS Companion for Microsoft Windows and Microsoft Access manuals. It seemed that a simple filename statement was all that was required. Although the DDE facility is documented, it took quite a bit of trial and error to successfully execute. The documentation defines the filename syntax as:

FILENAME fileref DDE ‘DDE-triplet’ | ‘CLIPBOARD’ <DDE-options>;

The challenge was to determine the DDE-triplet which unfortunately is application dependent. It is composed of:

- Application-name the executable filename of the server application.
- Topic the topic of conversation (between SAS and the DDE server application). This is typically the full path filename of the data you want to share.
- Item the range of conversation specified between the client and server applications.

The example in the documentation uses a Microsoft Excel spreadsheet. The manual suggests looking up the triplet in your application’s documentation or alternatively copying the selected data to the Clipboard so the Clipboard could identify it. Unfortunately our server application did not expose the DDE triplet to the Clipboard. The application documentation was sketchy as well. It did provide some required information but never clearly explained the DDE triplet. We experimented with code until we achieved success.

The code in Example 1 illustrates the Filename, Infile and Input statements that read data from our Microsoft Access database, Study1, into the SAS dataset, Patdemo. Our example uses macro code so accessing current data from the database could be easily repeated in any calling program. The DDE triplet maps to the Microsoft Access database where:

- Application-name = MSAccess
- Topic = Q:\Access\data\Study1.MDB (full path filename)
- Item = Table PatDemo!Data (entire Patdemo table)

The macro code specification imports the data from the database without the field names. There are many other options allowing the user to import field names, specific pieces of data or information about the data.

As with any application, care must be taken to properly address missing data. In Example 1, take note of the options we needed on the INFILE statement in order for the Access data to be processed correctly by SAS:

- dlm=’09’x indicates that the Tab (Hex=09) is the field delimiter.
- Notab tells the program not to convert tabs to blanks.
- dsd tells the program that two delimiters in a row signify a blank field.
- missover allows the expected variables in a line to be missing, instead of looking for them on the next line.

We created a %MAKEBOOL macro to convert boolean-type variables to absolute values. This was necessary because the Access Yes/No datattype transfers a Yes value as a -1 and a No value as a 0. Similarly, a %MAKEDATE macro was used to parse the mm/dd/yy Access date format, which is transferred as a character string, and create a SAS date.

Donna Usavage, Merck Research Labs. Merck & Co., Inc., West Point PA
Ellen L. Asam, Merck Research Labs. Merck & Co., Inc., West Point PA
Example 1: Macro to Establish DDE Link

```sas
%Macro Patdemo(dataset,dropkeep,subset);
********************************************************************
Macros Called:
    Makedate.sas called to convert dates read in as character to SAS dates.
    Makebool.sas called to convert Yes/No character field to boolean.
Macro Parameters:
    &dataset=SAS dataset name created
    &dropkeep=SAS Drop or Keep statements (i.e., Keep Alloc Trtgrp)
    &subset=any valid SAS Data step statement (i.e., If Trtgrp=1)
Sample Macro Call:
    %Patdemo(Patdemo,%STR(Keep Alloc Trtgrp),);
********************************************************************;
****  Establish DDE link to MSAccess Study1 database ****
****  PatDemo table.                                 ****;
filename patdemo dde 'MSAccess\Q:\Access\data\Study1.mdb;Table PatDemo!Data';
****  Read in data from PatDemo database table.      ****;
data &dataset;
infile patdemo dlm='09'x notab dsd missover;
&dropkeep;
input     alloc $
dobin $
gender $
race $
trtgrp
testrxc $
tolass
reasdisc $
.
.
%makedate(dobin,dob);
%makebool(testrxc,testrx);
run;
%Mend Patdemo;
LIMITING TRANSFERRED DATA
In our case, importing an entire Microsoft Access database table was desirable. However, DDE has the capability to import the results of a Microsoft Access query or the results of an SQL Statement. We have presented example syntax of each below. Example 2 uses the Microsoft Access query PatQuery, which was defined on the database Study1. Example 3 uses SQL code to limit data to specific records in the PatDemo table based on the result of the TolerAssess field.

Example 2: Using an Access Query

```sas
****  Establish DDE link to MS Access Study1 database ****
****  PatQuery table using an SQL select statement. ****;
filename patdemo dde 'MSAccess\Q:\Access\data\Study1.mdb;SQL Select * from PatDemo where TolerAssess=1;!Data';

LIMITATIONS
We found the major disadvantage of this method to be procedural. The Microsoft Access database being accessed MUST be open when the DDE request is made. It can, however, be minimized in the background. If the database is not open, or if the “Ignore DDE Requests” option is active, the DDE request fails, giving you the SAS error message: 
ERROR: Physical file does not exist.

EXTENDING THIS METHOD TO OTHER APPLICATIONS
Documentation is more readily available for reading data from and sending data to Microsoft Excel spreadsheets. However, we have included a very basic example below showing how to read and write to a spreadsheet. We observed that an embedded blank row in MSExcel will generate a SAS record with all blank values. However, trailing entirely blank rows in MSExcel will NOT generate a SAS record. Caution must be exercised when writing character data to a spreadsheet. Use of the tab character and the NOTAB file option are necessary when writing character data with embedded blanks.

Example 4: Reading Data from and Writing Data to MSExcel

```sas
****  Establish DDE link to MS Excel spreadsheet ****;
filename DDEFILE dde 'Excel\C:\phsug\ddetest.xls\Sheet1!R1C1:R4C5';
**** Read data in from spreadsheet ****;
data ddedata;
infile DDEFILE dlm='09'x notab dsd missover;
input A $ B $ C $ D $ E $;
**** Data manipulation section ****;
**** Write data to spreadsheet ****;
data _null_;
file DDEFILE notab;
set ddedata;
put A $ '09'x B $ '09'x C $ '09'x D $ '09'x E $;
CONCLUSION
In conclusion, utilizing DDE (Dynamic Data Exchange) to access Microsoft Access data from the SAS system proved to be very useful in our case. This method provides direct access to current data, can access data selectively (utilizing queries and SQL) and can be extended to other Microsoft applications (i.e., Microsoft Excel) if the DDE-triplet can be obtained. The SAS/Access method should also be considered by those who have the product licensed. Depending on the intended audience for an application, the requirement to have the database open during the DDE request may make this method undesirable.
For those who choose to utilize DDE from within SAS, the technical details included herein should prove to be helpful.
REFERENCES


SAS and SAS/ACCESS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

You may contact the authors at:

Ellen Asam  
Merck & Co. (BL3-2)  
West Point, PA 19486-0004  
(610)397-2664  
Email:  ellen_asam@merck.com

Donna Usavage  
Merck & Co. (BL3-2)  
West Point, PA 19486-0004  
(610)397-3011  
Email:  donna_usavage@merck.com