

A PROGRAM TO AUTOMATICALLY CONVERT MS ACCESS TABLES TO SAS DATASETS USING DDE

Rob Workman

INTRODUCTION

One way to import data from MS Access to SAS is to use DDE. DDE is very fast, and requires only Base SAS licensing. This paper will present a program to automatically convert an entire MS Access database or selected tables into SAS datasets. The result is one SAS dataset for each MS Access table that is converted. The user does not need to have any prior knowledge of the MS Access table or variable layout for this program to run. The program assumes the reader is running Windows 95, SAS version 6.12, and MS Access 97.

DDE SYNTAX

This program can import data from tables or queries previously set up in MS Access, or alternatively, an SQL statement can be included in the INFILE statement to generate a new query. The syntax for the DDE TRIPLET on the INFILE statement for all three query methods are in the table below.

PROGRAM NAME	DATABASE NAME	TOPIC	TABLE, QUERY, OR SQL STRING	DATA ITEM
MSACCESS	solution.mdb;	TABLE	example topics	!DATA
MSACCESS	solution.mdb;	QUERY	example topic list	!DATA
MSACCESS	solution.mdb;	SQL	select [topic id] from [example topics]	!DATA

To set up an INFILE statement, use the table above to generate the DDE TRIPLET used for the file location and add the options DEVICE = DDE, TRUNCOVER, DSD, NOTAB, and DLM = '09'X. Below are some examples of the INFILE statement for each of the options.

```
INFILE 'MSACCESS|solution.mdb;TABLE example topics !DATA' DEVICE = DDE
      TRUNCOVER DSD NOTAB DLM = '09'X;

INFILE 'MSACCESS|solution.mdb;QUERY example topic list !DATA' DEVICE = DDE
      TRUNCOVER DSD NOTAB DLM = '09'X;

INFILE 'MSACCESS|solution.mdb;SQL select [topic id] from [example topics]!DATA'
      DEVICE = DDE TRUNCOVER DSD NOTAB DLM = '09'X;
```

When the author used the sql text method, errors were generated when a "select *" statement was submitted instead of naming the variables individually, but the INFILE statement seems to work well when you name the variables in the SQL statement. Also notice that when using table names or variable names with embedded spaces in a select clause, you must wrap them in square brackets.

TABLE NAMES AND FIELD NAMES

To automate the procedure, the program automatically generates a list of the different table names, field names, and field types. MS Access does not place the same restrictions on naming conventions for fields and tables as SAS. Therefore you also have to convert those names into usable SAS names. In version 6.12, you have only 8 characters for each name. The program handles this by mapping only the first 8 characters of the table or variable names to the SAS names and using the original name as a variable label. In case of duplicates, the last character of the variable name will be assigned a number that increments for each duplicate. MS Access provides a DATA ITEM to retrieve the table names called "!TABLELIST". Here's an example of some code you could use to generate a list of tables.

```
data _null_;  
  infile 'MSACCESS|solution.mdb!TABLELIST' dsd notab dlm = '09'x  
  input;  
  put _infile_;  
run;
```

There are also DATA ITEMS to produce a list of variable names and types and the number of fields in each table. These are called "!FIELDNAMES;T" and "!FIELDCOUNT". These can be used in a data step similar to the one above by just replacing "!TABLELIST" with the new DATA ITEM.

PROBLEM

There is one situation where DDE will not work correctly. If within a text field there is a line feed character, SAS will interpret this as an end of record character and stop processing that observation. Then it will start with the character immediately following the line feed character. If the possibility of these characters exist within your data, watch to make sure that the number of observations in the dataset matches the number of observations in the MS Access table.

THE PROGRAM

To use the program, simply set the macro variable &DBNAME to the path and filename of your database and the macro variable &ACCSPATH to the path and filename of your access executable. Then submit the code. The program will convert all tables to sas datasets. You will have to modify the program if you want to convert queries instead of tables. There is also a place (see the comments in the program) where you can add a where clause to select specific tables to convert instead of the entire database.

CONCLUSION

This program should make short work of any MS Access to SAS conversions. It's automated so that it can be run without the attendance of a programmer. The code can be customized for special conversions that require only a few tables from the database.

REFERENCES

SAS Institute, TS325 *"The SAS System and DDE"*,
<http://ftp.sas.com/techsup/download/technote/ts325.pdf>

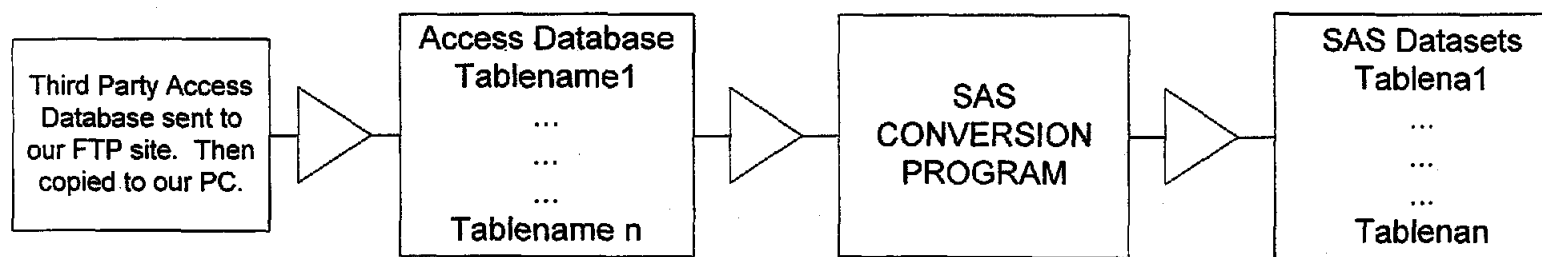
MS Access Help Menu, *"Use Microsoft Access as a DDE Server"*

AUTHOR CONTACT INFORMATION

Rob Workman
Sulzer Carbomedics
1300 E Anderson LN
Austin, TX 78752

(512) 435-3456
rworkman@carbomedics.com

Converting Access Tables to SAS Datasets



```

*-----*
*-- ACCDDE.SAS --*
*-- Programmer: Rob Workman --*
*-- Date: 06-30-99 --*
*-- Description: --*
*-- This program converts Access --*
*-- tables into SAS datasets using --*
*-- DDE. --*
*-- The table names and variable --*
*-- names will be converted to --*
*-- valid sas names and the --*
*-- datasets will be stored in the --*
*-- work directory. --*
*-- Be sure that the two macro --*
*-- variables defined at the top --*
*-- of the program match the --*
*-- correct name and path of your --*
*-- files. --*
*-- Note that currency variables --*
*-- and date/time values are being --*
*-- brought in as character data. --*
*-----*

```

```

/* dbname is the name and path of the database */
/* that you want to import into sas. */
/* accspath is the name and path of your */
/* MSACCESS.EXE file. */

```

* Change the next 2 lines *;

```

***
***
*****
***
*
;
%let dbname=p:\access\mydb.mdb;
%let accspath=C:\Program Files\Microsoft Office\Office\MSACCESS.EXE;

*
***
*****
***
***;

```

Specify the name of the
Access database here.

Start MS Access and wait until
Access finishes loading.

```
x ""&accspath"" ""&dbname"";

data _null_;
  rc=sleep(30);
run;

/* Get a list of tables */

filename tables dde "msaccess|&dbname!tablelist";

data tbls;
  length table $200 tablenm $8;
  infile tables lrecl=1048576 dsd notab dlm='09'x;
  input table $ @@;
  filevar="msaccess|&dbname;table "||trim(left(table))||
    '!fieldnames;T';
  filevct="msaccess|&dbname;table "||trim(left(table))||
    '!fieldcount';

  /* If the table name is not greater than 8 then */
  /* we don't want to change the trailing blanks */
  /* to underscores. */

  if length(table) gt 8 then
    tablenm=translate(substr(table, 1, 8), '_____',
      ' /!@#$$%^&*()-+=\|-{}[]:;<>,.');
  else tablenm=translate(trim(left(table)), '_____',
    ' /!@#$$%^&*()-+=\|-{}[]:;<>,.');

  if indexc(lowercase(tablenm),
    'abcdefghijklmnopqrstuvwxyz') ne 1 then
    substr(tablenm, 1, 1) = '_';

run;

proc sort data=tbls;

  *-- This makes a nice spot to put --*;
  *-- a where clause if you want --*;
  *-- to just import specific tables.--*;
  *--- For example: --*;

  * where table in ('del');

  by tablenm;
run;

/* Get a list of variables */

data fldnames fldtype(keep=type) ;
  length fldname $200 varnm $8;
  set tbls;
  by tablenm;
```

Convert the table names into valid SAS
table names.

```

/* assign a unique table name */
/* 8 characters long. */

if not first.tablenm then
do;
  if length(tablenm) eq 8 then
    substr(tablenm, length(tablenm),1)=put(varlast, 1.);
  else substr(tablenm, length(tablenm)+1,1)=put(varlast, 1.);
  varlast+1;
end;
else
  varlast=1;

/* Get a count of the variables */

infile dummy1 filevar=filevct device=dde lrecl=1048576 dsd notab dlm='09'x;
input fldcnt;

/* Get the variable names */

infile dummy2 filevar=filevar device=dde lrecl=1048576 dsd notab dlm='09'x;
do i=1 to fldcnt;
  input fldname $ @;

/* Create an 8 character field name with no blanks. */
/* Blanks are converted to underscore characters. */

  if length(fldname) gt 8 then
    varnm=translate(substr(fldname, 1, 8), '_____ ',
                     '/!@#$$%^&*()-=+\|~{}[]:;?<>,.'');
  else
    varnm=translate(trim(left(fldname)), '_____ ',
                     '/!@#$$%^&*()-=+\|~{}[]:;?<>,.'');

  if indexc(lowercase((varnm)),
    'abcdefghijklmnopqrstuvwxyz') ne 1 then
    substr(varnm, 1, 1) = '_';

/* This will be used to sort the dataset in the */
/* original order later in the program, so that */
/* we can set up the input statements correctly. */

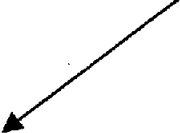
  sortvar+1;
  output fldnames;
end;
do i=1 to fldcnt;
  input fldtype @;

/* Determine the variable type. */
/* Data type 5 is Currency and data type 8 is */
/* date/time values. */

  if fldtype >=10 or fldtype in (5,8) then type='C';

```

If truncating past 8 characters creates duplicate table names, add a unique digit to the name.



Convert the variable names into valid SAS variable names.



```

else type='N';

output fldtype;
end;

drop i varlast fldtype;
run;

data fldnms;
  set fldnames;
  set fldtype;
run;

proc sort data=fldnms;
  by table varnm;
run;

/* eliminate duplicate variable names */
/* (names that are the same when */
/* truncated to 8 characters.      */

data fldnm;
  retain tablenm;
  set fldnms;
  by table varnm;

  if first.table then
    do;
      varlast=1;
    end;

    /* Create a unique variable name that */
    /* is only 8 characters.                */

  if not first.varnm then
    do;
      if length(varnm) eq 8 then
        substr(varnm,8,1)=put(varlast, 1.);
      else substr(varnm, length(varnm), 1)=put(varlast, 1.);
      varlast+1;
    end;
  drop varlast;
run;

/* Sort the dataset in the original order so that */
/* the input statements will list the variables */
/* in the correct order.                          */

proc sort data=fldnm out=fldnm(drop=sortvar);
  by sortvar;
run;

/* Create the data step code to include later.    */

```

If truncating past 8 characters creates duplicate variable names, add a unique digit to the name.



```

filename code catalog 'work.access.code.source';
data _null_;
  set fldnm;
  by table notsorted;
  file code;

  array tblvars (1000) $ 8 _temporary_;
  array varstype (1000) $ 1 _temporary_;
  array varlbl (1000) $ 200 _temporary_;

  if first.table then
  do;
    put 'data ' tablenm '(label='' table '')';
    put " infile 'msaccess|&dbname; table "
        table 'data''/' device=dde trunccover '
        'dlim=''09''x notab dsd;';
    i=0;
  end;

  i+1;
  tblvars(i)=varnm;
  varstype(i)=type;
  varlbl(i)=fldname;

  if last.table then
  do;

    /* Create a length statement */
    /* Character variables (and dates/times) */
    /* will have a length of $200. Numeric */
    /* variables will have a length of 8. */

    put ' length ';
    do j=1 to i;
      if varstype(j)='C' then put
        tblvars(j) ' $200 ';
      else put tblvars(j) ' 8 ';
    end;
    put ' ';

    /* Create the input statement. */

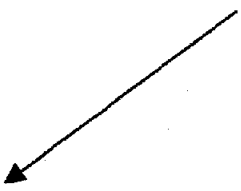
    put ' input ';
    do j=1 to i;
      if varstype(j)='C' then put
        tblvars(j) ' $ ';
      else put tblvars(j) ' ';
    end;
    put ' ';

    /* Create a label statement. */

    put ' label ';
    do j=1 to i;
      put tblvars(j) ' = '' varlbl(j) ''';
    end;
  end;

```

Generate the datastep code to perform the conversion.



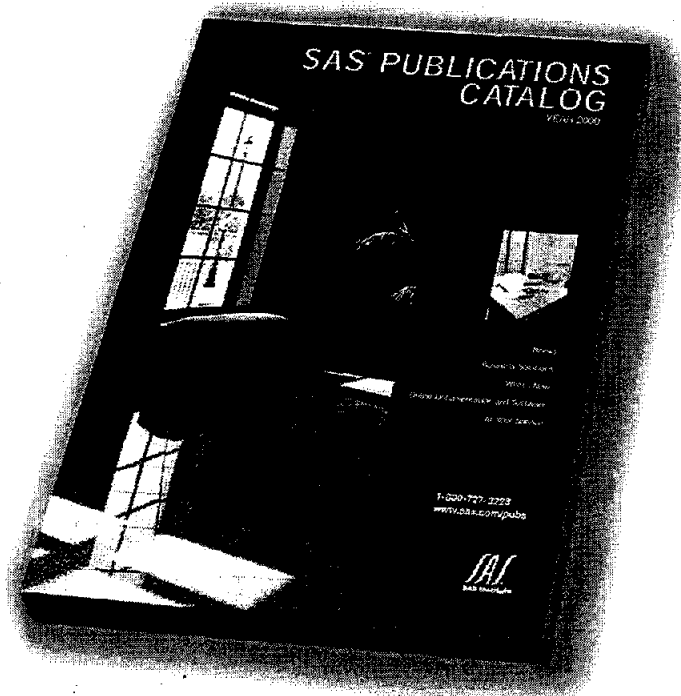
```
end;  
put ';;'  
  
put 'run;'  
put;  
end;  
run;  
  
/* submit the code created in the */  
/* previous step to create the    */  
/* tables in the work directory. */  
  
%inc code;  
  
***** end of program *****;
```

Run the code that was
generated in the previous
step.



Save **15%** on SAS Institute books while at the conference.*

Visit the SAS® Publishing Booth to learn more about...



- **New Books**
- **Books by Users™** - users wrote them, users love them
- **SAS Online Tutor™** - web-based training contains 35 lessons that teach how to use version 8 SAS environment
- **SAS® Certified Professional Training Guide, V6** - contains instruction designed to prepare you for the SAS Certified Professional V6 Exam (A00-001)

SAS Publishing offers you a comprehensive library of documentation for each component of the SAS Software System. Stop by the booth, place your order and receive a **15% discount** on your SAS documentation purchase. SAS Institute staff will be on hand to answer any questions and to assist you with your order.

Enjoy the conference!

sas.
e-Intelligence

SAS Publishing

The Power to Know™

SAS and all other SAS Institute Inc. product and service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates a USA registration.
* Certain restrictions apply. Please note that this offer does not apply to SAS Institute specialty items, SelectText, or the video-based training courses. No other discounts apply.
37723.0800



Announcing the First-Ever Joint Conference of the
South-Central SAS Users Group and
the SouthEast SAS Users Group.

August 19-22, 2001

InterContinental Hotel

New Orleans, Louisiana

Co-Chairs: Debbie Buck & Dave Riba

<http://www.sesug.org/ssu2001/welcome.htm>

