MOVING VMS® DATA TO MS-EXCEL®: USING SAS/CONNECT® AND DDE TO DOWNLOAD DATA FROM A VMS/ALPHA® NETWORK TO EXCEL SPREADSHEETS ON A PC

Randall Cates, Arbor Consulting Resources, St. Louis, MO.

Introduction

I was given the task of moving a large amount of data from a VMS/Alpha network to Microsoft-Excel spreadsheets. With this presentation I will demonstrate how this was accomplished; using SAS on a PC, connecting to a SAS session on the VMS/VAX using SAS/CONNECT, downloading the data to the local SAS session, and then finally with DDE commands, creating the Excel spreadsheets. I will also show you some quirks of the data and the process which made this particular task interesting.

The Problem

The data consisted of a number of separate studies existing in two separate formats: RDB® tables accessed from within SAS by Proc SQL statements and SAS datasets. All of the studies had to be separately transferred to Excel spreadsheets for transport to another company.

There were two groups of studies, each study needing multiple spreadsheets to be created. There was one set of 5 studies which had 20 tables per study, and another set of 9 studies which had 16 tables per study all totaling over 200 separate spreadsheets.

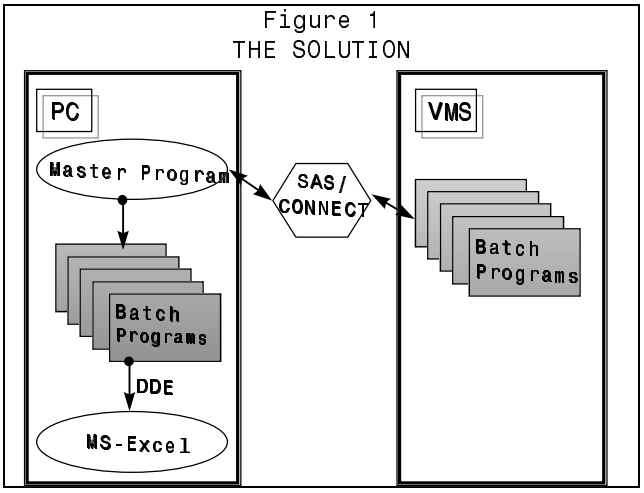
However working to our advantage we had the following:

- all studies within the groups needed the same types of spreadsheets.
- There was a standard naming convention across all studies.
- Each record had as primary control vars: study type, study id, and subject id.
- The data had already been thoroughly analyzed and reports generated using VMS/SAS. These reports closely matched the proposed contents of the Excel spreadsheets. So we could take those programs and modify them to produce transfer datasets.

The Solution

Here is how we broke down the process:

Figure 1
THE SOLUTION



From the PC,

1) Start a SAS/Connect session to the VMS/ALPHA network.
2) Create transfer datasets in the VMS/SAS session.
3) Download the transfer datasets to the PC then disconnect the SAS/Connect session.
4) Start an Excel session and connect to it using DDE.
5) Download the data to the Spreadsheets.
6) Format the spreadsheets, save them and quit the Excel session.

To do this we used three separate sets of programs:

1) **Batch programs** on the VMS/ALPHA side that created temporary transfer datasets. These were essentially copies of the original reporting programs modified for the purpose. They accessed the data using Macro

statements, Proc SQL views, and data statements.

2) **Batch programs** on the PC side that; started Excel, used DDE statements to download the data to the Spreadsheets, format the Spreadsheets, and finally saved them and quit Excel.

3) **Master Control Programs** on the PC side that; started SAS/CONNECT sessions, called the VMS/SAS batch programs, downloaded the data, closed the SAS/CONNECT sessions, and called the PC/SAS batch programs.

SAS/CONNECT ADVANTAGES

SAS/CONNECT software gives you more effective use of your different systems by allowing you to combine the different strengths of each system. Here are some ways that SAS/CONNECT can help.

Remote Processing of data;

With SAS/CONNECT you can send the processing of data to the most appropriate part of your system. For example, you can develop your SAS programs using sample datasets on your local PC, then move them to your UNIX workstation for processing your larger datasets.

Ease of File Transfer

You can easily and transparently upload or download SAS datasets, SAS catalogs, and even external files. You can back up any or all of your files to a remote host.

Improved Graphics Processing

You can use the graphics capabilities of your local host while using the processing power of your remote host. Perhaps you have a better plotter or printer attached to your local machine but your data resides on a more powerful remote host. With SAS/CONNECT you can run a graphics program on the remote host and then display the result in your local SAS session.

Multiple Connections

A big advantage of SAS/CONNECT is the ability to open multiple remote sessions. Now you can transparently combine data from what once might have been incompatible systems. The SAS/CONNECT manual offers the example of accessing an ORACLE® database on one system and an Rdb/VMS database on another then combining the data on your local host.

Automated Processing

You can use SAS/CONNECT in most any of the possible SAS formats; line mode, SAS Display Manager Windows, and batch programming (which is how we used it).

Step 1 -- Start a SAS/Connect session to the VMS/Alpha network.

Figure 2
Start a SAS/CONNECT Session

```
options comamid=tcp remote=vaxpams;

signon 'sasroot\saslink\tcpvms.scr';

OR

filename rlink 'sasroot\saslink\tcpvms.scr';
signon rlink;

rsubmit;
```

SAS/CONNECT offers different protocol options for connections; TCP/IP, DECNET, etc. depending on your system. We used the TCP/IP protocol. To start the SAS/CONNECT session you need an external file containing special SAS commands that initiate the link. SAS provides many of these script files and you can tailor them to your needs. For our system using TCP/IP, the file was tcpvms.scr. This file does a number of things when called. It:

- invokes the software programs necessary for the different hosts to talk to one another,
- logs on to the remote host, and
- optionally, queries the user for logon id and password.

You also need an options statement with two options; 'commamid=' and

'remote='. 'Commamid' tells SAS what type of communications method we are using. 'Remote' is the name of the remote host that we want to access.

The next command 'signon' creates the link and starts a SAS session on the remote host. To do this we tell SAS the name of the external script file and where it is located on the local host. You can also place this information into a filename statement and access that as in our second example. At this point, SAS comes back to you with query windows asking for your logon id and password. Type them in exactly as you do when you log on normally.

The final command 'rsubmit' submits any statements following to the remote SAS session.

One quirk I found with SAS/CONNECT and VMS/ALPHA is that when I had a customized prompt (for example 'RCCATES:') rather than VMS standard '$' prompt, SAS would hang. SAS seems to read the $ prompt as a signal of a proper connection.

Step 2 -- Call VMS/SAS batch programs to create temporary datasets.

Now comes some simple but extremely powerful programming. With judicious use of macro variables, macro calls, and generic programs we were able to set up a programming assembly line for creating the transfer datasets.

Figure 3
CREATE TEMPORARY DATASETS

```
%let stdynum=A1;
%let stdytyp=PURPLE;

filename getdat1 'rccate sample\xxx.x';
%include getdat1;
run;

filename getdat2 'rccate sample\xxx.x';
%include getdat2;
run;
```

The first statements created macro variables in the remote SAS session. These macro variables defined the study types that we wanted to access. Next the 'filename' statements defined SAS batch programs on the remote host. Finally, the '%include' statements ran the programs.

The beauty of this programming is twofold. One, we were able to string together programs to create all the necessary transfer datasets for one particular study number and type in one remote SAS session. Two, since within study groups all studies were virtually the same, we could just redefine the macro variables for a new study and rerun the control program.

Step 3 -- Download the temporary datasets to the PC then disconnect SAS/CONNECT session.

Figure 4
DOWNLOAD DATA

```
PROC DOWNLOAD DATA=VAXDAT1 OUT=PCDAT1;
RUN

PROC DOWNLOAD DATA=VAXDAT2 OUT=PCDAT2;
RUN

endrsubmit;

signoff;
```

Once all the transfer datasets were created, each one was individually downloaded to the local SAS session using SAS/CONNECT commands. Note that we used similar naming conventions from the remote SAS session to the local SAS session.

The last step with using SAS/CONNECT was to close the remote session. To do this we first stop submitting commands to the remote SAS session by the 'endrsubmit' command. 'signoff' then ends the remote SAS session and disconnects the link.

Now we had all of the temporary work datasets for a particular study type and study on the PC. To transfer this data to Excel, we called the PC/SAS programs.

DDE

DDE stands for Dynamic Data Exchange. It is a powerful tool for transferring data between applications. DDE has been compared to a conversation between two people. The exchange starts when one person/application asks a question of the other. Then the questioner stops talking and waits for an answer.

DDE is termed Dynamic because both applications are running at the same time and 'talking' to each other and the processes of one of the applications (server) are controlled by the requests from the other (client).

In our example, SAS is the client and Excel is the server.

Call PC/SAS batch programs.

Figure 5
CALL PC/SAS BATCH PROGRAMS

```
%let study=%str('[formula('STUDY A1
%let titl1 =
  %str('[formula("DEMOGRAPHIC DATA"
%let name =
  %str('[save.as("c:\sasample\
filename print1 'c:\sasample\
%include print1;
run;
```

The code for the transfer of data to the spreadsheets was written in separate PC/SAS programs that the master program now called. Again we set up macro variables to control the output. These are the statements that we would change when rerunning the programs for a different study.

The code for calling PC/SAS programs is the same, a filename statement, and a %include macro command. The next steps took place within the batch programs. Each program was tailored for each type of transfer dataset.

Step 4 -- Start an Excel session and connect to it using DDE.

Figure 6
START MS-EXCEL AND CONNECT

```
options noxwait noxsync;
x 'c:\msoffice\excel\excel.exe';

data _NULL_;
   x=sleep(5);
run;

filename PCEX1 dde 'excel|sheet1!r1c1
  NOTAB;
```

We started out with an options statement. The options 'noxwait noxsync' refers to the way SAS deals with the Windows environment and essentially tell SAS to Not wait for whatever you start with external commands.

The 'X' command sends a call out to Windows to start an Excel session.

Now here's a conundrum. You want SAS to continue working, but normally, when you use the X command SAS waits for whatever was started to finish before it continues. If we start Excel in this way, SAS would not move to the next step until we shut down Excel, defeating the purpose. That's why we added the options so that SAS would continue to the next step.

Now the other side of the coin is that Excel requires some amount of time to start up. If SAS starts sending DDE requests to Excel before it is up and running, the DDE requests will fail. So we built in a finite time gap with the 'data _null_' statements. I found out that with my PC, 5 seconds was sufficient.

Once Excel is up and running we tell SAS to connect with a filename statement, specifically defining a DDE link and telling SAS where to look within Excel. The quoted name is the 'DDE Triplet' which defines what application, what part of the application, and what section is accessed.

Step 5 -- Download the data to the Spreadsheets.

Figure 7
INSERT DATA TO SPREADSHEETS

```
OPTIONS MISSING='.';

data _null_;
  set PCDAT1 end=finis;
  by study patient testdate;
  file PCEX1 ls=410;
  put      study     5.0       '09'x
           patient   3.0       '09'x
           testdate  mmddyy8.  '09'x
           (more vars)
           wgt       8.0       '09'x  ;
```

Now we could finally transfer the data to the Spreadsheet. We did this using 'Data _Null_' steps and put statements. We output the dataset to the defined DDE link, where SAS requested Excel to input the data in the format defined by the put statement. Excel recognizes the '09'x word as a Tab character which it needs to keep the data points in their proper columns.

Step 6 -- Further format the spreadsheets (optional), save the spreadsheets and shut down the Excel session.

Figure 8
FORMAT SPREADSHEETS

```
filename cmds dde 'excel|system';

data _null_;
  file cmds;
  put '[formula("STUDY","R8C1")]';
  put '[formula("PATIENT","R8C2")]';
  put '[formula("TEST","R7C3")]';
  put '[formula("DATE","R8C3")]';

  put '[column.width("C1:C10",9)]';
```

At this point, we had transferred the data to an Excel spreadsheet in an Excel session. We had not yet saved the data to a permanent spreadsheet however. We also wanted to format the spreadsheet to be easily understandable by those at the receiving end. To do this, we needed another separate 'filename' statement and 'data null' statement.

We re-connected to the Excel session with the filename statement. This time instead of defining a section of the spreadsheet with a DDE Triplet, we connected to the 'Excel|system' which gave us formatting capabilities for the Spreadsheet.

Then it was just a matter of 'putting' a series of formatting statements, in Excel Macro language, which we obtained from the Excel manuals. The first four statements are examples of adding column headers. The next statement reformats the readable width of the columns to the widest cells.

The next two statements are an example of using macro variables. We had set these macro variables in the control program to uniquely identify the study that the data came from. These particular macro variables inserted titles.

Finally, the last statements saved the data to a permanent Excel file and closed the Excel session. This last statement was necessary to clear the way for the next batch program.

Conclusion

Transferring data between SAS and other software packages and also between computer systems is an increasingly common activity for computer programmers. With this presentation, I have tried to demonstrate how we SAS gurus can 'get the job done' within SAS by the use of such tools as SAS/CONNECT and DDE, do it with style and panache, and, hopefully, really impress the client.

Requirements:

On PC

- Version 6.10 Base SAS® for the PC.
- SAS/CONNECT® Software for PC/SAS Release 6.10.
- TCP/IP access set up to VMS/Alpha network with user id and logon password.
- Microsoft Windows® 3.1 or Windows 95®.
- Microsoft Excel® Version 4.0.

On VMS/Alpha Network

An account on the network that has access to SAS.
Version 6.09 Base SAS® for VMS.
SAS/CONNECT® Software for VMS/SAS Release 6.09.
Version 6.11 VMS®.
Version 1.3a of VMS/Ultrix® Connection.
Standard '$' prompt.

Technical Sources:

Books;

SAS Language Reference, Version 6.

SAS Companion for the Microsoft Windows Environment, Version 6.

SAS/CONNECT Software, Usage and Reference Version 6.

SAS Technical Report P-223, SAS/CONNECT Software: Changes and Enhancements for the CMS, MVS, and VMS Environments, Release 6.07.

Microsoft EXCEL, User's Guide 1, User's Guide 2, and Function Reference, Version 4.0.

Other Sources;

SAS WWW Site: http://www.sas.com, SAS NOTES.

Bodt, Mark (1996), "Talking to PC Applications Using Dynamic Data Exchange" Observations: The Technical Journal for SAS Software Users, 5(3), 18-28.