

A Case Study in Deploying SAS Programs on Unix Using Windows Based Tools

Samuel T. Croker, Lexington, SC

ABSTRACT

This case study explores using existing SAS programs, an x-windows telnet client and a Microsoft Access based interface to deploy complex applications requiring user interaction when AF, Intrnet or FSP are not available. In this case the x-windows-telnet client is used programmatically to build a bridge between a Microsoft Access front-end and the SAS-DB2 server. The data is stored in DB2 and returned to the user via linked Access tables. Since no data is stored in Access, the front end is small in size and can be deployed over a network easily. Topics covered are building the Windows to Unix bridge, dealing with security issues, retaining users' logs and output, building the user interface and providing SAS interactivity using the WINDOW and %WINDOW commands.

INTRODUCTION

This is a worst-case scenario in which there is only Base SAS on Unix, an x-windows client and Microsoft Office. For this case study, Microsoft Access will be used. There are many better ways of deploying this type of program and they should all be looked at before attempting this method.

PARTS OF THE SYSTEM

Interactivity is limited when only BASE SAS is licensed but there are a variety of ways that SAS programs can be executed remotely, and there are several ways that communication with the user may occur. Here I the entry of program parameters using SAS Macro Windowing.

THE RDBMS

Another assumption is that a rdbms is present. The database facilitates the viewing of remote data in Access through an ODBC or similar connection.

THE SAS SERVER

There is a SAS server installed on the UNIX machine.

THE PC

All we have is a pc with Microsoft Access and an x-windows emulator. For this study the x-client/server is Hummingbird Exceed. Exceed has additional functionality (scriptable telnet, security settings etc...) that makes this process a little easier. Any scriptable telnet client can be used in a similar manner.

EXAMPLE: MICROSOFT ACCESS AND EXCEED

BUILDING THE EXCEED BATCH FILE

The file that will be used to start a remote SAS session is created on the fly using an Access module. Exceed has a module called XSTART that allows the automatic connection between a client and a remote Unix server via XWindows. The XSTART interpreter handles all of the settings between the remote server and the PC, including user authentication. The *.xs file that is created below is a list of settings in text format and can be read using any text viewer. It is useful to first create an XStart file using the interface provided by Exceed, and then examine this file using a text editor. Each line of the XSTART file would be placed in the function below as *script line 1* etc...

This is the general idea for how this is done without actually describing the exact process. The exact process varies considerably among systems and clients.

Example:

```
Public Sub RunXwin(commandstring as string)
    Dim executionstring, fname as string
    Fname = "filepath\tempfilename.xls"
    Open fname for output as #1
    Print #1 "script line 1"
    Print #1 "script line 2"
    ...
    Close #1
    Executionstring = "filepath\XStart.exe " & " " & fname & " "
    Call Shell(executionstring,1)
```

End Sub

The Command line for the XSTART file should look something like:

```
cd //path_to_sas_programs/; sas -dms sasprog.sas
```

The `-dms` or `-nodms` environment variable prevents the generation of all of the standard Display Manager windows. The only windows generated will be the windowing environment that is coded, and the Session Control SAS window.

THE SAS SERVER SIDE

The components of the SAS server side of access deployable tools are:

- ☐ A windowing environment such as data step or macro windows
- ☐ Log and output capture
- ☐ Error checking
- ☐ Display of results

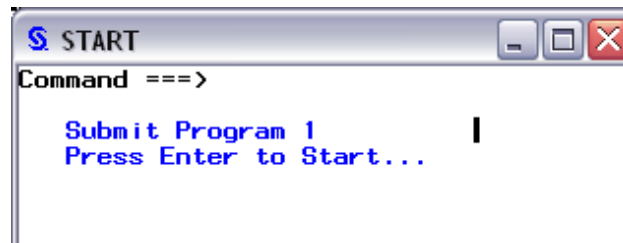
The great thing about building the SAS side of a windows deployable application is that it can be used by any program at all. There is no need to even use a program to call the SAS code as it could run all by itself.

SAS WINDOWING

Interactivity is the key when calling SAS code through a xwindows bridge. The following macros can be used to announce and report conditions resulting from the remote submission of code. Using macro windows is a nice way to make standard SAS code somewhat interactive.

The ANNOUNCESTART macro simply produces a user interaction element showing that processing is about to begin on the server.

```
%macro announcestart(title);  
  %WINDOW start ROWS=10 COLUMNS=40 IROW=10 ICOLUMN=10 COLOR=WHITE  
  #1 "    &titler"                color=blue protect=yes  
  #2 "    Press Enter to Start..." color=blue protect=yes #2 @30 que 1;  
  %display start delete;  
%mend announcestart;  
  
%announcestart(Submit Program 1);
```



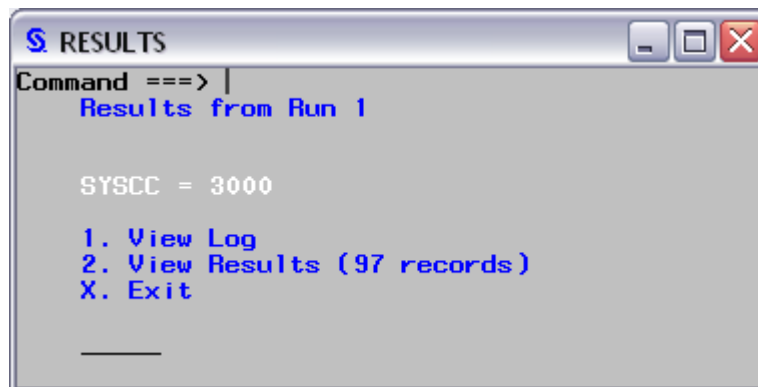
When the program completes, it is important to let the user know what happened.

```
%macro announceresults(title,recs,logfile,dataset);  
  %rewind:  
  %WINDOW RESULTS ROWS=15 COLUMNS=46 IROW=10 ICOLUMN=10 COLOR=Grey  
  #1 @5 "&title"        color=blue protect=yes  
  #4 @5 "SYSCC = &syscc"          color=white protect=yes  
  #6 @5 "1. View Log"            color=blue protect=yes  
  #7 @5 "2. View Results (%cmpres(&recs) records)" color=blue protect=yes  
  #8 @5 "X. Exit"                color=blue protect=yes  
  #10 @5 resp 5                  color=black attr=underline;  
  %display results delete;
```

```

%if &resp=1 %then %do;
    proc fslist file="&logfile";
    run;
%end;
%if &resp=2 %then %do;
    proc fsview data=&dataset;
    run;
%end;
%if %upcase(&resp)=X %then %goto ender;
%goto rewind;
%ender:
%mend announceresults;

```



All of these windows are wrapped together using a driver program in SAS like:

```

PROC PRINTTO
    LOG="&outpath/logfile.txt" new
    PRINT==&outpath/outputfile.txt= new;
RUN;
%announcestart(Begin SAS Program);
%PUT LOG DATETIME = %SYSFUNC(PUTN(%SYSFUNC(DATETIME()),DATETIME20.));
%let syscc=0;
<<PROGRAM STATEMENTS>>
PROC PRINTTO; RUN;
proc sql noprint;
    select count(*) into :recordcount from OUTPUT_TABLENAME;
quit;

%announceresults(Run
Complete,&recordcount,&outpath./logfile.txt,OUTPUT_TABLENAME);

```

There are a few factors that are very important in this wrapper program. In this example, the SAS program statements that are omitted but would go where the <<PROGRAM STATEMENTS>> are assumed to create or modify a dataset and the results are stored in a table (OUTPUT_TABLENAME) in the RDBMS.

Process of the wrapper:

1. Using PROC PRINTTO to capture logs and output. For this example there was no output, so only the log was captured.
 - a. If the log needs to be open to everyone a CHMOD command might need to be issued through an X command

```
x "chmod 744 &outpath./logfile.txt=";
```

- b. If different users will be using the code simultaneously, each user needs to have a separate outpath. Otherwise the logging will have security or overwrite problems depending on how security is set on the files.
2. Use of the SYSCC automatic macro variable to capture any error codes generated. Only the last error code will be reported using this method, but any error code means that an error occurred and the specific problems can be seen by looking at the log.
3. The number of records that are in the result table (OUTPUT_TABLENAME) are stored in the macro variable *recordcount*.
4. The *announceresults* macro that was defined earlier is called with the appropriate title, record count, path to the log, and name of the rdbms.

The announceresults macro then will display a window such as at the top of the page that allows the user to

1. See the SYSCC error code (typically 1-4 mean a warning was generated and >4 means an error)
2. View the log in an PROC FSLIST window
3. View and possibly edit all of the records in the OUTPUT_TABLENAME table using a PROC FSVIEW window.

LINKING RDBMS TABLES TO ACCESS

Access is a very efficient way to look at RDBMS tables through linking tables. This is a very simple thing to do, but if the table needs to be edited there is a big consideration that must be made. Access requires that there be a unique key in the remote table for updates to be possible. If a key is not available, or it is specified wrong when the table is linked a "Recordset is not updatable" message will be generated by Access and the grid control will not accept any modifications, deletions or additions.

Another confusing occurrence with linked tables is the #DELETED RECORD message in an Access control. This often (but not exclusively) results from Access not being able to uniquely identify a record. This could mean that the unique key specified at link time is not actually unique. It could also mean that the underlying table has been changed while the ODBC query was executing.

CONCLUSION

This technique is far from efficient and I recommend that it be a last resort. There are many sources of errors and isolating them is sometimes difficult and very confusing. It is useful if a end-user application needs SAS access and the SAS resources available to the programmer are limited. If the application needs extensive SAS access interactivity it would be much better to use SAS/INTRNET or build the application using webAF elements.

There are also a number of free xstart programs out there that I have not tried. It may be that some work better than others.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Samuel T. Croker
252 Kyzer Road
Lexington, SC 29073
803 240-3805
scoyote@scoyote.net
www.scoyote.net

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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