Paper 197-26

Converting *CP World* Application to SAS V8: A Discussion of Issues and Hurdles Encountered

Fang Dong, Pfizer Global Research and Development, Ann Arbor, Michigan

Abstract

The *CP World* reporting system is a SAS-based multi-platform application that runs on a Windows NT Server and a Unix host. It makes extensive use of Base SAS, SAS Macros, SAS/ACCESS, SAS/Connect and SAS Output Delivery System. It is used to report early phase clinical trial data at the Ann Arbor Laboratories of Pfizer Global Research and Development. The system design and features are originally presented in SUGI 25. In converting the CP World application into SAS V8, our primary goal is to take advantage of the new Output Delivery System features. A secondary goal is to simplify SAS coding by taking advantage of long character variable values and explore assigning libname to oracle database versus use SQL Passthrough to extract data from oracle database. The strategy we used is a two-step approach. The first step is to migrate the application into SAS V8. The second step is to explore the new features in V8 such as the ODS, assigning libname to Oracle database and long character variable values. This paper discusses the issues and hurdles that we encountered converting the application into SAS V8.

Skill Level: Intermediate

A Brief Description Of The *CP WORLD* Application

The *Clin Pharm World* (*CP World*) application is a standard reporting system used in the Experimental Medicine Department at the Ann Arbor Laboratories, Pfizer Global Research and Development. It is used to produce listing and summary tables as well as data visualization for early phase clinical trial data. The *CP World* application is developed in SAS 6.12 and makes extensive use of Base SAS, SAS/CONNECT® and SAS Dynamic Data Exchange (DDE) technology. It also utilizes Visual Basic (VB) macros that run under Microsoft Excel. The Unix portion of the application uses SAS SQL PASSTHROUGH to extract data from an Oracle database, and performs the intensive data manipulation. The Windows NT portion of the application serves as the interface between the user and the Unix host. Additionally, Windows NT performs a data output 'step' using SAS DDE technology. The VB portion of the application is invoked with SAS DDE and allows great flexibility and customization for table formatting. The final output is stored in Excel and accessed via the Pfizer intranet by our clients (i.e., clinical scientists, physicians and medical writers). For detailed description of the application, please refer to SUGI 25 Proceedings, paper p242-25 by Dong et al [1].

Converting *CP WORLD* Application To SAS 8.0

With the arrival of SAS 8.0, we have the opportunity to take advantage of the many new features offered by SAS. In contemplating the conversion of *CP World* from SAS 6.12 to SAS 8.0, we adopted a two step approach. The first step is to simply migrate the application into a SAS 8.0 environment. This way, any necessary changes to update the application can easily be identified. Once the application is running in SAS 8.0, we moved to the second step, to explore various new features and incorporate them into the application.

Step 1: Migration *CP WORLD* Application To SAS 8.0

As mentioned earlier, the *CP World* application was developed in SAS 6.12 running on a NT workstation and a Unix host. In deploying SAS 8.0, the company adopted the Citrix MetaFrame technology. That is, the SAS 8.0 application will be running on a Citrix server and the users will be accessing the application from a Citrix Client. For the most part, this upgrade of host environment is seamless for the migration of our application. We did, however, noticed a slower response time in the DDE process.

The SAS Dynamic Data Exchange (DDE) technology is where the SAS application exchanges data dynamically with other Windows® application such as Excel. DDE is used to output final reports to Excel worksheets in our application. In the process, the application opens the Excel workbook and brings it to the front of the windows and one can see the reports being created by SAS visually. It takes less than a minute to create a long reports in the NT workstation environment while it consistently takes more than a few minutes to 10-20 minutes to create the same length reports in the Citrix Client-Server environment. We verified that this phenomenon is true for both SAS 6.12 and SAS 8.0. Further, we tested this in a computer physically located next to the Citrix server and ruled out the interference of the network traffic. Therefore, we conclude that this is not due to SAS 8.0, rather, it is an inherited nature of the Client-Server environment. What is interesting is that when we minimized the Excel window, the DDE process ran much faster. The gained efficiency may be due to the fact that when the Excel window is minimized, the network traffic to bring data from the server to refresh the client screen is eliminated.

Surprises

The syntax of %syslput has changed. Remember in SAS 6.12, unlike the %sysrput statement, which is a SAS function, the %syslput is a SAS supplied macro. In SAS 8.0, %syslput has been upgraded to a function and the syntax to pass a value of 999 to macro variable *ci* has changed from

%syslput(ci,999,remote=ROC) in 6.12

to

%syslput ci=999 in SAS 8.0.

Further there is a bug in SAS 8.0 that only 32 %syslput is allowed in each SAS/Connect session. Figure 1 is an error message one gets when there is more than 32 %syslput statements.

According to SAS Technical Help, this bug has been fixed in SAS 8.1. In SAS 8.0, the workaround that we engineered is to store the macro variable values in a data set, use proc upload to upload the data set to Unix and then use call Symput function to transfer the values back into macro variables. Figure 2 shows the sample code.

Figure 1

```
2    /*proc printto
log="c:\c_data\test.log";run;*/
3    %let ci=1008;
4    %syslput ci=999;
5    %syslput ci=999;
...
33   %syslput ci=999;
34   %syslput ci=999;
Illegal Instruction In Task [
Program  ]
Fault Occured at [UNKNOWN at
0x00000000]
```

Figure 2

```
data macvars1;
   length macvar $8 macval $8;
  macvar="ae_in"    ; macval=&ae_in;   output;
  macvar="demo_in"  ; macval=&demo_in; output;
  macvar="dose_in"  ; macval=&dose_in; output;
  ….
Run;

 rsubmit;
  proc upload inlib=work outlib=work;
   select macvars1;
  run;

 data _null_;
  set macvars1;
  call sympút(macvar,trim(macval));
 run;
```

The last surprise comes from how the Dynamic Library Link (DLL) call works. It does not work the same way in Version 8 as it did in Version 6.12.

The FindWindowA routine is written in Windows API language and is saved in the file findw.txt. This is a sascbtbl file and is shown in figure 3.

The FindWindowA routine tries to determine whether the Excel application is open in the current Windows session by retrieving the Excel handle.

The SAS program that calls the FindWindowA routine that works in 6.12 is shown in figure 4.

However, this will result in an error in SAS 8.0, as shown in figure 5.

Version 8 requires the class name of the window being searched - the first variable of the modulen call. With the assistance from SAS Technical Help, we modified the DLL call, figure 6 shows an example of finding an Excel window that works under Version 8.

Figure 3

```
Routine FindWindowA
   minarg=2
   maxarg=2
   stackpop=called
   module=USER32
   returns=ushort;
 arg 1 char input
format=$cstr200.;
 arg 2 char input
format=$cstr200.;
```

Figure 4

```
FILENAME sascbtbl
'c:\temp\findw.txt';

DATA _NULL_;
    /* Retrieve the handle of
the Excel window */
hwnd=MODULEN('*ie','FindWindowA'
,,'Microsoft Excel');
run;
```

Figure 5

```
ERROR:  Read Access Violation In
Task [ DATASTEP ]
Exception occurred at [65FD62CB]
Task Traceback
Address   Frame     (IMAGEHLP API
Version 4.0 rev 5)
65FD62CB  03BDF8F8  0001:000052CB
uwumodul.dll
65FD39FE  03BDFEA0  0001:000029FE
uwumodul.dll
65FD1318  03BDFEC8  0001:00000318
uwumodul.dll
```

Figure 6

```
FILENAME sascbtbl
'c:\temp\findw.txt';
DATA _NULL_;
    /* Retrieve the handle of
the Excel window */
  length class $200;
  class="XLMAIN";
hwnd=MODULEN('*ie','FindWindowA'
,class,'Microsoft Excel');
run;
```

Exploration of New Features

After overcoming the hurdles described above, the *CP World* application is up and running in SAS 8.0. Now is the time to explore the new features offered by SAS 8.0.

Long Character Values

The first thing we tried is the long character variable values. In Oracle Clinical database, the title of a clinical study can be several hundreds characters long. In SAS 6.12, we used the following code to split the long title into several smaller values (Figure 7).

Figure 7

```
select title1, title2, title3 into :stitle1,
:stitle2, :stitle3
      from connection to oracle
        (select substr(c.title,1,200) TITLE1,
               substr(c.title,201,400) TITLE2,
                substr(c.title,401,600) TITLE3
         from clinical_studies c
         where c.study = 1000);
```

Later on, we concatenate these smaller values and output them as a single title using code similar to what is shown in figure 8.

Figure 8

```
data _null_;
 file excel noprint;
 set final;
    put
    %do c=1 %to 3;
        stitle&c
    %end;
  ;
run;
```

In SAS 8.0, the SAS code can be simplified to the following extracting and output blocks respectively (figure 9 and 10).

Figure 9

```
Extracting:
select title into:stitle
      from connection to oracle
        (select c.title TITLE
         from clinical_studies c
         where c.study = &_study);
```

Figure 10

```
Output:
data _null_;
 file excel noprint;
 set final;
    put  "&stitle";
run;
```

Direct Access to Oracle Database via Libname

Next we tried to access Oracle database by assigning libname directly to Oracle tables.

Here is an example,

libname test oracle user="ops$dongf" pass="*****" path="mypath" schema="S999_1$STABLE" dbmax_text=3000;

It will assign libname test to 'S999_1$STABLE' schema in the oracle database residing in 'mypath' . User is an authorized user name and pass is the password. It is important to include the parameter values in quotes since many long or special characters are acceptable to Oracle and yet confusing to SAS. In Oracle Clinical database, the views for each clinical study is in its own schema. In the above example, we are concerned about the stable view for the study 999_1. There are many other public schema that we can explore this way. However, if a schema is big, the response time is long. This is due to the fact that SAS will need to bring all the information to its own temporary work space. This is a nice way to explore the structure of an Oracle database, without using SQLPlus or other non-SAS products. However, we did not find any big benefits to switch the data extraction from Proc SQL PASSTHROUGH to directly assign libname to the Oracle database.

Output Delivery System

Keep in mind that our primary motivation in using SAS 8.0 is to take advantage of the new ODS features. Figure 11 is a piece of simple SAS code that can generate powerful visual presentation of clinical ECG data, shown in figure 12.

For an in depth discussion on how we expand the *CP World* application to present clinical trial data visually, please refer to Synowiec's SUGI 26 paper[2].

Conclusion

SAS 8.0 offers many enhancements in terms of data access and output delivery systems. In converting our standard reporting application, *CP World,* from SAS 6.12 to SAS 8.0, we encountered a few surprises, discovered a few bugs, and had fun exploring several new features. Overall, the upgrading from 6.12 to 8.0 is seamless in many aspects of SAS, such as SAS/Connect, SAS/Access, SAS/DDE. Looking to the future, we are anxious to test several other enhancements such as the RTF driver and PDF driver, which will be available with SAS 8.1 and 8.2. That will move SAS based reporting to a new era.

Figure 11

```
ods listing close;
goptions reset=all rotate=landscape
        device=gif transparency noborder;

filename odsout 'c:\temp';
ods html body='ecgplot.html' path=odsout;

   axis1  label=(h=1 a=90 'Mean Delta QTC (msec)');
   axis2  label=(h=1      'Time (Hours Post Dose)');
 proc gplot data=ecgtime;
   plot meanqtc *time=regtx/vaxis=axis1 haxis=axis2;
   Title1 h=1.5 "Mean Delta QTC FDA Method
Predose Baseline VS Time";
run;

ods html close;
ods listing;
```

REFERENCES:

1. Dong et al. (2000)
"A Standard Reporting System for Phase 1 Clinical Trials
A SAS® Based Application '*Clin Pharm World*' ". Proceedings of the Twenty-Fifth Annual SAS User Group International Conference, 25.

2. Synowiec, R (2001)
Data Visualization of Phase 1 Clinical Studies-"Big Al's World". Proceedings of the Twenty-Sixth Annual SAS User Group International Conference, 26.

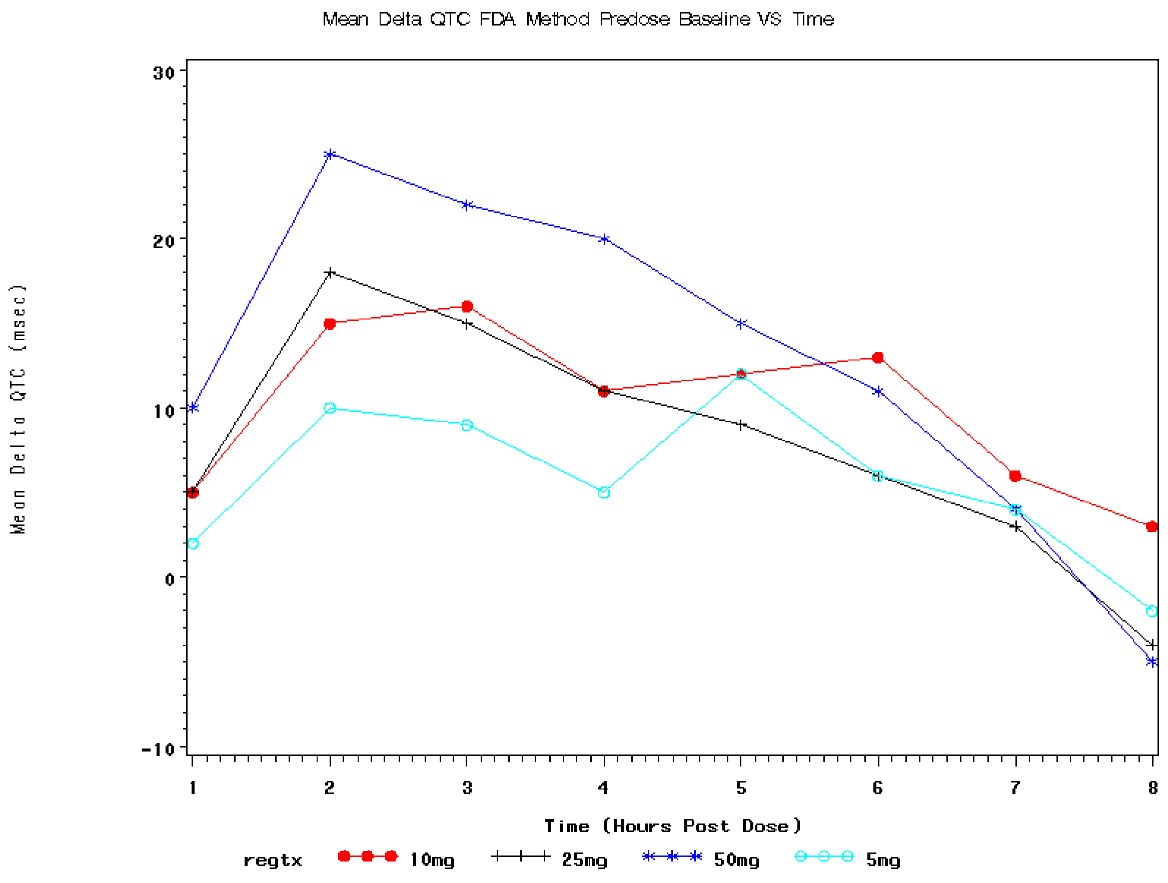
ACKNOWLEDGEMENTS:

The author wishes to thank John Arbuckle and many other colleagues for their support in this conversion endeavor.

CONTACT INFORMATION

Fang Dong
Ann Arbor Laboratories
Pfizer Global Research & Development
2800 Plymouth Road
Ann Arbor, MI 48105
Email: fang.dong@Pfizer.com

Figure 12


Mean Delta QTC FDA Method Predose Baseline VS Time
Mean Delta QTC (msec)
Time (Hours Post Dose)
regtx
10mg
25mg
50mg
5mg