The Realities of Downsizing: Moving a SAS® Application from MVS® to UNIX®

Kimberly J. LeBouton, K.J.L. Computing

Abstract

Downsizing in the current business world is deemed necessary for companies to remain competitive. Companies are downsizing mainframe applications to smaller platforms in the hope of saving money and gaining productivity for both users and developers. This paper will explore the realities of downsizing an MVS SAS application to UNIX at a major manufacturing company.

Managing the Project

- Business reasons for moving the SAS application to the UNIX platform.
- Key issues in planning and managing the transition.
  - What operating system to use for UNIX emulation: Windows NT® or Windows 95®?
  - Downsizing when the new platform is not yet in place.
- Convincing "mainframe dinosaurs" that UNIX is a productive world for SAS programmers.

Key Technical Issues

- Moving 20 gigs of data across platforms on a weekly basis.
- Managing over 300 datasets.
- Converting SAS code.

To conclude, a financial picture will be presented to show what gains were actually made in reducing mainframe CPU utilization.

Managing the Project

Why Move to UNIX?

The Service Engineering Information (SEI) department started as most end user computing groups do—not enough I.S. resources to fulfill user demands. COBOL reports could not be written fast enough to meet needs. Also, analysts wanted access to analytical tools in order to analyze their data. Thus, SAS was purchased and installed on MVS in 1985 to meet the unfulfilled needs of the user community. SAS datasets were built and end users were trained in the rudiments of SAS.

By 1990, SAS programmers used SAS/AF® to develop menu systems that create sophisticated analytical programs. The menus were used primarily by analysts to study warranty issues related to the product lines. The SAS programs developed from input by the analysts interleave many different datasets and use many SAS products, such as SAS/STAT®, SAS/ETS®, and SAS/GRAPH®. The programs were then run either interactively or in batch.

Due to the volume of data and the sophistication of the analysis, mainframe CPU and DASD utilization for the Service Engineering group was very high. The average mainframe CPU utilization was 2.8 hours per day. This accounted for fifty percent of the MIPS on the end-user mainframe side (i.e., LPARC). Twenty-seven GB of permanent storage and 2.5 GB temporary storage were required. The average queue time was about one hour and forty minutes, while average CPU usage was

about one minute. The users felt response time was often slow. Batch turnaround normally took a day, which did not satisfy emergency needs for information.

Service level on the mainframe had been degrading as more users wanted access to data. In addition, the atomic level of the data warehouse was being implemented on LPARC. To compound the problem, the decision was made to cease adding resources to the end users' machine. This led the SEI department to seriously consider if the current operating environment was the right solution for their computing needs.

Early in 1995, a study was conducted by SAS Consulting Services to explore resource usage, environment, and customer performance expectations.[1] Their conclusions were as follows: (1) off load cycles to improve performance, (2) move to a distributed environment, (3) create summary datasets and modify existing datasets to move to a LAN server; and (4) implement coding changes to improve efficiency.

These suggestions from SAS, and others proposed by the Information System Department (ISD), resulted in the development of a project to explore moving SEI processes from MVS/TSO® to a RS6000®/UNIX environment. This move was organized into two phases. The first phase, which began in late 1995, entailed a feasibility analysis of transferring SAS resources to the RS6000. Issues included cost efficiencies, resource utilization and performance, transportability of SAS code, and a comparison of data accessibility methods (i.e., SAS datasets, SAS views, and/or DB2® views). The findings were encouraging. SAS code conversion required minimal changes, and decreases in elapsed times were significant. I joined the project at the beginning of the second phase, which was the actual migration of SAS code and associated data to the RS6000. It was decided that both phases of this project would be independent of mainframe Data Warehouse implementation.

This paper has two objectives: (1) To identify issues that caused a schedule slippage despite the application of traditional project scheduling techniques, and (2) To develop prescriptive approaches to help minimize the impact of these issues on future projects.

Managing the Transition

Prior to my joining the project, a project schedule had been established with all of the known requisite components necessary to finalize the second phase. Several components, outside my control, were late. For example, it was planned that the three permanent SAS programmers would have access to UNIX beginning in March, but due to competing hardware and software delays, access was not available until September. In addition, the analysts did not have access to UNIX until late November. One of the programmers received a UNIX workstation to test in late July, but we faced a significant schedule variance by that time. Although the plan was for me to spend the majority of my time converting SAS code, more than sixty percent of my time was spent establishing methods to download the data on a regular basis. This is one of the key issues of downsizing, that is, the extent of effort necessary to transfer data was not completely understood at the planning stage.

Accessing UNIX

Much of the delay in getting the programmers onto UNIX was due to the fact that the approved PC operating system, Windows 3.1®, had difficulty handling the UNIX emulation software, eXceed®. The move to a 32 bit operating system was not planned to be made

until later in the year, but this project forced an early decision on this issue. We tested Win NT® and eXceed for Win 3.1 for over a month with very stable results, but Windows 95® was selected as the approved operating system. As mentioned above, a UNIX workstation was also tested, but the programmers did not want an additional terminal on their desk. The programmers and analysts also wanted to be able to continue to "cut-and-paste" SAS output into other software applications. The UNIX workstation could not handle this requirement, at this time. When the programmers were finally able to access UNIX, they were thrown a curve with the new operating system. In fact, they were faced with a new learning curve. This is another key issue in downsizing, the often overlooked task of researching, selecting and testing the operating system prior to project implementation. Alternatively, the task could have been to confirm the capability of the existing operating system to handle the emulation software.

New Platform Issues

Printing also took longer than planned. The status quo was to print out very large reports for corporate headquarters. This type of printing would be required of the new platform too. The schedule estimated that this printing capability would be in place by May, but it was not available until October. Throughout this delay, matters were further complicated when it was arbitrarily determined that the mainframe printer could not be used as a UNIX printer. In the end, the mainframe printer was able to be defined as a UNIX printer. The key issue here is to be suspect of simple "no" answers and to do your own research to determine whether the roadblock truly exists.

A high quality color printer was brought into the SEI department to be tested as a UNIX printer in September, although it was planned to be delivered in April. Another department had an identical printer since April, but it was not defined as a UNIX printer until late June. Prior to being defined, if you wanted to send UNIX output to this printer, you were required to use FTP code and a very long address. Although the other department's printer was available in June, it was not accessible to us because it was located in a secured area. Therefore, the schedule slipped another several months since our printer did not arrive until September. The key point here is to define the required hardware and factor in adequate lead time for delivery.

Printing to the mainframe printer and printing to the color printer required separate UNIX printing commands, with different printing options. To get the text printouts to match for both printers, time was spent determining which font would allow for the same printing options in SAS. Large sections of the AF programs referred to printing. This SAS code conversion was left undone until the printing environment was fully in place.

Convincing 'Mainframe Dinosaurs'

One of the original concerns of this project was the anticipated resistance of users to a new platform, but fortunately the mainframe become more sluggish. Response times became so bad that UNIX was now viewed as the promised land. Since the data was being continually refreshed, UNIX could be used for ad-hoc requests.

The SAS programmers were comfortable using the SAS Display Manager on the mainframe, which works similarly on the new platform. Also, the Windows-look of the RS6000 under AIX® was very familiar. Several years ago this group had training with SAS/AF FRAME, but were not able to exploit many of the features until they began to use SAS on the RS6000. The on-site UNIX support personnel were comfortable with UNIX commands, but the SAS

programmers are forcing them to learn more about the windowing nature of AIX. The sluggish mainframe, the familiarity of SAS Display Manager, and the windowing features of AIX resulted in little to no resistance to the new platform.

Technical Issues

Weekly Transfer of Data

The biggest single issue for this project was getting over twenty gigs of data down to the RS6000 on a weekly basis. As previously stated, this alone took over sixty percent of my time. The twenty gigs of data were comprised of 356 datasets, primarily SAS datasets. These datasets changed so much in a week that it was necessary to bring down a new copy every week. The data needed to be refreshed by start of business on Monday mornings.

Three different approaches were tried before we determined the best transfer method. The first approach, which was conducted involved testing SAS/CONNECT® for over a month to see if we could get the data down by Monday morning. The connection for SAS/CONNECT originated on the RS6000, but we were only able to transfer the data on late Saturday night or early Monday mornings, because of maintenance to the mainframe on Sundays. The transfer speed using SAS/CONNECT was too slow to work within the stated constraints.

We also had connection problems. Programs were created to restart, if the connection broke, but the mainframe account would get locked up for hours. We were unsure what was causing this problem, SAS, RS6000 or MVS, but we needed to get the data down. It was concluded that SAS/CONNECT would not be able to handle the demands of such a large, frequent transfer.

Aside: As suggested in the Hardy, Barrios, and Muller paper (see reference section), SAS/CONNECT has never been seriously tested in the opposite direction. That is, have SAS/CONNECT originate on MVS and push the data to the RS6000. This will be explored when dealing with the smaller daily downloads of data.

The second approach was to try to use the FILENAME FTP transfer method that became available with version 6.11 of base SAS. Had we been able to use SAS/CONNECT, we would not have needed holding tanks for our data on both platforms, which we knew was a requirement of PROC CPORT and PROC CIMPORT. FILENAME FTP only required a holding area on MVS to get the data down. Unfortunately, we have had problems with this method, and still have an outstanding tracking number with SAS Technical support.

The final approach was to create production jobs using PROC CPORT, and then FTP the file down to the RS6000. A SAS program was developed on the RS6000 to grab the transported file and turn it into a SAS dataset with PROC CIMPORT. The SAS dataset also needed to be resorted, because of the EBCDIC to ASCII conversion. See Appendix I for an example of the four steps used to get the data down.

Because we had spent so much time working on the first two approaches, we were really pressured to get the data down fast. Working within the constraints of not processing on Sundays and also wanting to be sure that the data update was complete on MVS, we started eight production jobs under two separate job streams on Monday mornings at 2:00AM. The jobs would finish around 10:30AM on Mondays. It took over three hours of CPU time to CPORT the data and over 8 hours of I/O processing to transfer the data down to the RS6000.

Successful downloads became a reality in late July.

To cut transfer rates in half, hopes were placed on CLIO/S®—an IBM® data transfer product. CLIO/S, which was scheduled to be installed by May for another RS6000 project, was not installed until late September. Due to some irregular UNIX installation, CLIO/S never performed as was hoped. Because of this, we needed to modify the production jobs to get the data transferred early on Mondays. We were also having some problems with our current production jobs. The datasets grow in size each week as the year continues. Some of the larger datasets were causing the production jobs to fail, because of space limitations. Due to CLIO/S not working as planned, and the space limitation, the eight production jobs were rewritten into 72 production jobs. All but one of the production jobs were able to follow the production jobs that created the MVS datasets. The six largest datasets, that accounted for over 20 percent of the 20 gigs of transferred data, were immediately sent to tape after the completion of their respective production job.

Managing Over 300 Datasets

The many programs that were needed to transfer the datasets were easier to create and maintain because of the "pipe" option on FILENAME under UNIX. This option, combined with SAS DATA _NULL_ report writing, was used to create the JCL for the production jobs and over six hundred separate FTP transfer programs. See Appendix II for code used to create the FTP programs.

Converting SAS Code

Because of the portability of SAS code, the conversion of SAS code was minimal. It was decided that the SAS code was to work both on MVS and UNIX. Modified SAS autoexecs are shown in Appendix III for MVS and UNIX. Also shown are examples of the SCL code to support the dual platform approach. The key issue here is that what was originally thought to be a SAS programming effort turned out to be a macro-level project management effort.

Conclusion

To summarize, I hope that what I presented in this paper will serve as a guide to those who will be involved with downsizing an application. I have tried to point out areas that caused schedule slippage and also provide remedies to these problem areas. Adding tasks to a traditional timeline may be the only remedy, but it is better to anticipate the problems before they arise.

Since the transfer of data is such a timely issue, I want to readdress it one last time. Whether it be related to transferring information via the Internet, refreshing a data warehouse, or transferring data between two different computer platforms, the technology has not caught up with the high demand for the need to move data quickly. Make sure that you have planned the requisite steps necessary to have the data available for use on the new platform.

Although financial gains are not yet available from the completion of this project, the following is known. The downsizing added some additional CPU time on the production mainframe and also required additional DASD. The analysts will not be completely removed from the mainframe, but their CPU usage on the end user machine will be significantly less.

Other Technical Resources

The following Internet mail lists, with their respective addresses, were used during this project:

SAS-L
listserv@uga.cc.uga.edu
AIX-L
listserv@pucc.princeton.edu
EXCEED-L
exceedusers-request@hummingbird.com

Acknowledgments

A special thanks to those who responded to my many postings on SAS-L concerning this project, especially Tim Berryhill, Tim Latendress, Greg Barnes Nelson, and Karsten Self.

Thanks to Tracy Cermack who was instrumental in getting this project started, reviewed this paper for content, and for coining the term "mainframe dinosaurs."

Also, thanks goes to my husband, Al LeBouton, who was my editor, and also found numerous activities for our sons to do, while I worked on this paper and presentation.

Notes

[1]The unpublished study is titled, Technical Services Engineering and Auto Warranty SAS® End-User Computing Study by SAS® Consulting Services Inc. for American Honda Motor Company, and was given to the company on June 9, 1995.

References

Andrais, B. and Kirtz, J. (1994), *Moving off the Mainframe: Downsizing and Cost Reductions.* Proceedings of the Nineteenth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 643-653.

Carr, R., and Bretheim, D. (1995), *The Other Five Percent: Writing SAS Code for Multiple Operating Systems.* Proceedings of the Twentieth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 94-99.

Cohn, D. (1994), *An AIX® Companion.* Englewood Cliffs: Prentice-Hall, Inc.

Fischell, T. (1993), *SAS® Transport Files or SAS/CONNECT® Software: Considerations for the Proper Choice.* Proceedings of the Eighteenth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 1243-1250.

Grippo, K., Chen, J. and Brown, L. (1995), *Building a Data Warehouse with SAS® Software in the UNIX® Environment.* Proceedings of the Twentieth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 519-524.

Hardy, K., Barrios, A., and Muller Sally (1996), *You Want Me to Move How Many Thousand Files from MVS to UNIX?* Proceedings of the Twenty First Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 1611-1621.

Hoffman, D. (1996), *The Push to "Get Off of the Mainframe" and Move to Client/Server: Is the Software Industry Keeping Up With*

the Demand for Data Storage and Access? Proceedings of the Twenty First Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 1473-1478.

Lindsey, M. (1996), *Cross Platform Generation of Reports from Oracle Using SAS® on MVS, Macintosh and AIX*. Proceedings of the Twenty First Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 718-723.

Nelson, G.B. (1995), *The SAS® System UNIX Primer.* Proceedings of the Twentieth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 1416-1419.

Plath, D. (1994), *Downsizing a Mainframe SAS® Application to UNIX: An Application Developer's Perspective.* Proceedings of the Nineteenth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 438-441.

SAS Institute Inc. (1993), *SAS Companion for UNIX Environment: Language,* Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1993), *SAS Companion for UNIX Environment: User Interfaces,* Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc.,(1990), *SAS Companion for the UNIX Environment and Derivatives,* Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1995), *SAS Software: Changes and Enhancements, Release 6.11,* Cary, NC: SAS Institute Inc.

Scott, S. (1993), *Why We Replaced DB2® Software with SAS® Software as a Relational DBMS for a 30-Gigabyte User Information System.* Proceedings of the Eighteenth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 187-196.

Wayte, L. (1996), *SAS® to the Rescue! A UNIX® System Administrator's Tool.* Proceedings of the Twenty First Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 1451-1456.

Witmer, B. and Dower, D. (1995), *Mainframe to Windows™ in a Snap: Moving a SAS® Application Among Operating Platforms.* Proceedings of the Twentieth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 1348-1352.

Young, M.L. and Levine, J. (1995), *UNIX® for Dummies®.* Foster City, CA: IDG Books Worldwide, Inc.

About the Author

Kim LeBouton is an independent consultant with 14 years experience with SAS. Her areas of expertise include base SAS, SAS/STAT, SAS/FSP®, and SAS/AF software. She has a BA degree in Psychology from California State University, Long Beach and a MA degree in Educational Statistics from UCLA. Kim has been selected as WUSS 1997 Co-Chair. She has recently applied to become a SAS Institute Quality Partner.

Kim LeBouton
K.J.L. Computing
3431 Yellowtail Drive
Rossmoor, CA 90720
(562) 594-9235
email: Kim_LeBouton@msn.com

Appendix I: Transfer Program

Main Program

```
//PCZ02J01  JOB 01CCZ,'LABOR TRANSFER      ',CLASS=A,MSGCLASS=M,
//             TIME=1440,REGION=4096K
//*
//JOBLIB  DD  DSN=SYS3.LINKLIB,DISP=SHR
//*
//CALL001  EXEC SASSYS,CONFIG='CB18.MXG.USERLIB.BKUP(CONFIGK1)'
//ICPLIB   DD  DSN=ICP.SAS.LABOR,DISP=SHR
//XPORT    DD  DSN=HM.CZ99AB.D01.SAS.LABOR.XPORT(+1),
//         DISP=(NEW,CATLG,DELETE),
//         RECFM=FB,
//         LRECL=80,
//         SPACE=(80,(2,1),RLSE),
//         AVGREC=M,
//         UNIT=DISK
//SYSIN    DD  DSN=SYS3.SAS.PROGRAMS(CZ99XP),DISP=SHR
//*
//CALL002  EXEC CZ99AB,
//         MEMBER=C0201X01,
//         NODE4=SAS,
//         NODE5=LABOR
//*
//CALL003  EXEC SASSYS,CONFIG='CB18.MXG.USERLIB.BKUP(CONFIGK1)'
//FTPRUN   DD  DSN=HM.CZ99AB.D01.SAS.LABOR.FTP(+1),
//         DISP=(OLD,KEEP,KEEP)
//SYSIN    DD  DSN=SYS3.SAS.PROGRAMS(CZ99FT),DISP=SHR
//*
//CALL004  EXEC CZ99AC,MEMBER=C0201F01
//*
```

Step 1 SAS Program (CZ99XP)

```
PROC CPORT LIBRARY=ICPLIB FILE=XPORT;
RUN;
```

Step 2 PROCLIB (CZ99AB)

```
//CZ99AB   PROC MEMBER=MEMBER,NODE4=NODE4,NODE5=NODE5
//*
0/CZ99AB05 EXEC PGM=FTP
//SYSPRINT DD  DSN=HM.CZ99AB.D01.&NODE4..&NODE5..FTP(+1),
//             DISP=(NEW,CATLG,DELETE),
//             RECFM=FBA,LRECL=80,
//             SPACE=(80,(10,10),RLSE),AVGREC=K,
//             UNIT=DISK
//INPUT    DD  DSN=SYS3.OPTNCTL(&MEMBER),DISP=SHR
```

FTP Program (C0201X01)

```
RS6000 Name
UNIX Account
UNIX Password
locsite autor di
put 'hm.cz99ab.d01.sas.labor.xport(0)' /appl/work/sas.labor.xport
quit
```

Step 3 SAS Program (CZ99FT)

```
 DATA FTP(KEEP=MSG) CLEAN(KEEP=FTPCLEAN);
   INFILE FTPRUN PAD END=LAST;
   INPUT MSG $133.;
   /* ADDITIONAL MESSAGES MAY NEED TO BE ADDED AS MORE */
   /* POTENTIAL ERRORS OF FTP ARE FOUND                */
   IF INDEX(UPCASE(MSG),'ABEND') OR
      INDEX(UPCASE(MSG),'ABORT') OR
      INDEX(UPCASE(MSG),'ERROR') OR
      INDEX(UPCASE(MSG),'FILE NOT FOUND') THEN NOTCLEAN+1;
   OUTPUT FTP;
   IF LAST THEN DO;
      IF NOTCLEAN>1 THEN FTPCLEAN='N';
      ELSE FTPCLEAN='Y';
      OUTPUT CLEAN;
   END;
RUN;

  /* DETERMINE IF CLEAN TRANSPOSE OCCURED */
  /* THE TWO SETS ARE REQUIRED IN CASE IT */
  /* WAS NOT A CLEAN RUN AND THE MSGS NEED*/
  /* TO BE ADDED                          */
DATA COMPARE;
   IF _N_=1 THEN SET CLEAN;
   SET FTP;
   FILE FTPRUN;
   IF FTPCLEAN='N' THEN DO;
      IF _N_=1 THEN PUT @1 'N' @10 "&SYSDATE" @20 "&SYSTIME";
      PUT MSG;
   END;
   ELSE DO;
      PUT @1 'Y' @10 "&SYSDATE" @20 "&SYSTIME";
      STOP;
   END;
RUN;
```

Step 4 PROCLIB (CZ99AC)

```
//CZ99AC  PROC MEMBER=MEMBER
//*
//CZ99AC05 EXEC PGM=FTP
//SYSPRINT DD  SYSOUT=*
//INPUT    DD  DSN=SYS3.OPTNCTL(&MEMBER),DISP=SHR
```

FTP Program (C0201F01)

```
RS6000 Name
UNIX Account
UNIX Password
locsite autor di
put 'hm.cz99ab.d01.sas.labor.ftp(0)'   /appl/work/sas.labor.ftp
quit
```

Appendix II Program Creation

```
filename units pipe 'ls -d /appl/icp/icp.sas.*/';
data minijcl;
   infile units pad;
   input entire $35.;
   if index(entire,'cust.') or index(entire,'labor') or
index(entire,'model') or
      index(entire,'parts');
run;
data minijcl;
   set minijcl;
   typdat='A';
   typdatn='01';
   jobname=compress('PCZ99J'!!typdatn);
   rmicp=substr(entire,11);
   icpdsn=compress(rmicp,'/');
   rmpref=substr(entire,19);
   dsn=compress(rmpref,'/');
     /* MVS Data Sets */
   uicpdsn=upcase(icpdsn);
   udsn=upcase(dsn);
   node4=scan(udsn,1);
   node5=scan(udsn,2);
   if node5=' ' then do;
      node5=node4;
      node4='SAS';
   end;
   node4=trim(node4);
   hpdsn=compress('HH.CZ99AB.D01.'!!node4!!'.'!!node5);
   uxptdsn=trim(hpdsn)!!left('.XPORT');
   uxptdsn1=trim(hpdsn)!!left('.XPORT(+1)');
   uftpdsn=trim(hpdsn)!!left('.FTP');
   uftpdsn1=trim(hpdsn)!!left('.FTP(+1)');
   xportmem=compress('CZ99'!!typdat!!'X'!!put(_n_,z2.0));
   ftpmem=compress('CZ99'!!typdat!!'F'!!put(_n_,z2.0));
   file '$HOME/pcz99j01.sas';
   if _n_=1 then put
"//" JOBNAME  " JOB 01CCU,'OTHER FILE TRANSFER
',CLASS=A,MSGCLASS=M,"
/'//             TIME=1440'
/'//*'
/'//JOBLIB  DD  DSN=SYS3.LINKLIB,DISP=SHR';
   step+1;
   put
'//*'
/'//CALL' step z3.0 '  EXEC SASSYS'
/'//ICPLIB   DD  DSN=' uicpdsn +(-1) ',DISP=SHR'
/'//XPORT    DD  DSN=' uxptdsn1 +(-1) ','
/'//         DISP=(NEW,CATLG,DELETE),'
/'//         RECFM=FB,'
/'//         LRECL=80,'
/'//         SPACE=(80,(500,100),RLSE),'
/'//         AVGREC=M,'
/'//         UNIT=DISK'
/'//SYSIN    DD  DSN=SYS3.SAS.PROGRAMS(CZ99XP),DISP=SHR'
/'//*';
   step+1;
   put
 '//CALL' step z3.0 '  EXEC TCZ99AB,'
/'//         MEMBER=' xportmem +(-1) ','
/'//         NODE4=' node4 +(-1) ','
/'//         NODE5=' node5
/'//*';
   step+1;
   put
 '//CALL' step z3.0 '  EXEC SASSYS'
/'//FTPRUN   DD  DSN=' uftpdsn1 +(-1) ','
/'//         DISP=(OLD,KEEP,KEEP)'
/'//SYSIN    DD  DSN=SYS3.SAS.PROGRAMS(CZ99FT),DISP=SHR'
/'//*';
   step+1;
   put
'//CALL' step z3.0 '  EXEC TCZ99AC,MEMBER=' ftpmem;
run;

proc fslist file='$HOME/pcz99j01.sas';
run;
```

Appendix III SAS Code Conversion

UNIX Autoexec

```
  /*****************************************/
  /**       MVS/UNIX CONVERSION             **/
  /** TO PROVIDE A MORE GLOBAL ACCESS TO    **/
  /** LIBNAMES/FILENAMES THE FOLLOWING      **/
  /** VARIABLES HAVE BEEN CREATED.          **/
  /** SYSTEM    $4  TOGGLE BETWEEN MVS/UNIX **/
  /** PERMROOT  $20 UNIX NEEDS ROOT DIRECTORY**/
  /**               AND MVS DOESN'T. USED   **/
  /**               ICP FOR MVS, SINCE ''   **/
  /**               PUT IN AN EXTRA SPACE TO **/
  /**               MVS FILE TO RESOLVE     **/
  /** access    $20 allows for disp=shr      */
  /** wait      $20 allows for wait=30       */
  /** wait5     $20 allows for wait=5        */
  /*****************************************/

data _null_;
   call symput('iddlc',lowcase(sysget('USER')));
run;

options ls=132 pagesize=59;
options printcmd="enscript";
dm 'formname default';

   %let access  =;
   %let afsave  =/appl/sei/&iddlc/afsave;
   %let annos
=/appl/sei/utility_files/sas_datasets/map_datasets;
   %let ctryds  =/appl/sei/utility_files/sas_datasets;
   %let fmtlib  =fmtlib;
   %let graphv6 =/appl/sei/&iddlc/menu_system_output/graphics;
   %let ib04root=/appl/icp/icp;
   %let libnfmt =/appl/sei/utility_files/sas_format_library;
   %let ossystem=UNIX;
   %let outstuff=/u/ia06/tempsave.sasdata;
   %let permroot=/appl/icp/icp;
   %let wait    =;
   %let wait5   =;

options pagesize=500;
libname  sei6    '/appl/sei/cb04/sas_programs/seimenu';
filename out     '$HOME/sasout.data';
filename batch   '$HOME/batch.sas';
libname  library '/appl/sei/utility_files/sas_format_library/';
libname  afsave  "/appl/sei/&iddlc/afsave";
```

MVS Autoexec

```
 /*****************************************/
 /**       MVS/UNIX CONVERSION             **/
 /** TO PROVIDE A MORE GLOBAL ACCESS TO    **/
 /** LIBNAMES/FILENAMES THE FOLLOWING      **/
 /** VARIABLES HAVE BEEN CREATED.          **/
 /** SYSTEM    $4  TOGGLE BETWEEN MVS/UNIX **/
 /** PERMROOT  $20 UNIX NEEDS ROOT DIRECTORY**/
 /**               AND MVS DOESN'T. USED   **/
 /**               ICP FOR MVS, SINCE ''   **/
 /**               PUT IN AN EXTRA SPACE TO **/
 /**               MVS FILE TO RESOLVE     **/
 /** ACCESS    $20 allows for disp=shr      */
 /** wait      $20 allows for wait=30       */
 /** wait5     $20 allows for wait=5        */
 /*****************************************/
   %let access=disp=shr;
   %let afsave=.afsave.sasdata;
   %let annos=ia06.annotate.sasdata;
   %let ctryds=ia06.ctryds.sasdata;
   %let fmtlib=fmtlib;
   %let graphv6=.graphv6.sasdata;
   %let libnfmt=cb04.format.library;
   %let ossystem=mvs;
   %let outstuff=ia06.tempsave.sasdata;
   %let permroot=icp;
   %let wait=wait=30;
   %let wait5=wait=5;

LIBNAME LIBRARY 'CB04.FORMAT.LIBRARY' DISP=SHR;
DM 'AF C=SEI6.MAINCAT.MAIN.MENU NOBORDER=YES' CONTINUE;
```

SCL Code

```
INIT:

   LENGTH SYSTEM $4
          LIBNFMT PERMROOT CTRYDS AFSAVEDS GRAPHV6 $60
          ACCESS RUNTIME $8
          PRT WAIT WAIT5 $7
          CTRYSTRG $200;

   CONTROL ASIS;

   ACCESS=SYMGET('ACCESS');      /* SYSTEM VARIABLE DEFINITION */
   CTRYDS=SYMGET('CTRYDS');      /* SEE CORRESPONDING AUTOEXEC */
   AFSAVEDS=SYMGET('AFSAVE');    /* FOR DESCRIPTION            */
   GRAPHV6=SYMGET('GRAPHV6');
   LIBNFMT=SYMGET('LIBNFMT');
   PERMROOT=SYMGET('PERMROOT');
   SYSTEM=SYMGET('OSSYSTEM');
   WAIT=SYMGET('WAIT');
   WAIT5=SYMGET('WAIT5');

   IF SYSTEM=UPCASE('MVS') THEN DO;
      CALL EXECCMDI('GCURSOR OFF');
      IDD=SYMGET('SYSJOBID');
   END;

   IF SYSTEM=UPCASE('UNIX') THEN DO;
      IDD=SYSGET('USER');
      IDDLC=LOWCASE(SYSGET('USER'));
   END;



   /* CALL PRINT ROUTINE */
   PRTFMT=COMPRESS('$'!!OSSYSTEM!!'PRT.');
   PRT=PUTC(IDD,PRTFMT);
   NAME=PUTC(UPCASE(IDD),'$SEIFMT.');

   /* OPENING DATASET THAT STORES SCREEN SELECTIONS */
   RC=LIBNAME('AFSAVE',AFSAVEDS);
   DSID=OPEN('AFSAVE.WARRSALE','U');
   CALL SET(DSID);

RETURN;
```