Ferreting Out Year 2000 Compliance Problems With PROC SOURCE

Michael A. Raithel, U.S. Customs

Abstract

As the new millennium rapidly approaches, numerous organizations have intensified their efforts on Year 2000 compliance projects. Many organizations have allocated significant staff time and financial resources to ensure that they find and fix their non-Year 2000 compliant applications. Predictably, a number of vendors are taking advantage of this new focus by providing packages that help to identify problematic code. However, IS shops running SAS software under OS/390® (MVS®) on mainframe servers already possess a powerful tool that can be used in Year 2000 conversion projects.

PROC SOURCE is a Base SAS® Software tool that can process OS/390 Partitioned Data Sets (PDS). Coupled with SAS character handling functions, PROC SOURCE can become a very powerful tool for examining PDS's for non-Year 2000 compliant constructs.

This paper demonstrates how to use PROC SOURCE to process PDS's that contain source code, load modules and production Job Control Language (JCL).

Introduction

In most organizations, the bulk of the Year 2000 compliance problems can be found in the vast inventory of production COBOL programs. These programs are the workhorses of organizations and perform tasks ranging from data entry, to accounting, to inventory, to payroll. For years, COBOL programmers cleverly dropped the two century digits of the year portion of date fields to make their programs more efficient. They did not know that while they were saving two bytes of storage, they were unwittingly helping to fan the flames of a major turn-of-the-century programming problem. Now, organizations are spending millions of dollars and countless work-hours to put the two century digits back into COBOL programs.

COBOL programmers normally create and store their source code in members of a "source library" PDS. They compile and link their source programs into load modules that are stored in members of a "load library" PDS. Programmers also store their production Job Control Language--which is used to execute their load library COBOL programs--in members of a "production JCL library" PDS. Consequently, Partitioned Data Sets play a major role in the creation, storage and execution of COBOL programs.

All of this is fortunate, because PROC SOURCE is a SAS procedure that is designed to process PDS's. PROC SOURCE can be used to create a file that contains the directory information (including the name of each member) of a PDS. It can also be used to create a file that contains the contents (each line) of each member of a PDS. Both of these functions are the basic building blocks of systematic methods for searching for Year 2000 compliance problems. By judiciously coupling these functions with SAS language character manipulation features and with SAS procedures, you have a powerful tool set for identifying a number of Year 2000 compliance problems.

This paper contains three examples of how you can use PROC SOURCE to ferret out Year 2000 compliance problems. The following three sections describe a specific Year 2000 compliance issue, and how it can be addressed via one of the SAS code examples. To facilitate a better flow of the prose, the SAS code for all of the examples is located at the end of the paper. All three of the SAS programs have been simplified and heavily commented to promote their readability. You are invited to use these examples as a junping-off point for exploring the Year 2000 compliance issues in your own organization.

Matching Source Programs with Load Modules

One of the first tasks most organizations undertake in a Year 2000 project is to take an inventory of source/load library matches. That is: they determine if they have the source code for all of their load library programs. Since the source library programs and load library programs (usually) have the same names, the inventory is simply a matter of matching a listing of the source library PDS members against a listing of the load library PDS members. PROC SOURCE is ideal for facilitating this task.

In **Example 1**, the members of PROD.SOURCLIB are matched with the members of PROD.LOADLIB to determine:

1. Which members of PROD.SOURCLIB and PROD.LOADLIB match
2. Which members of PROD.SOURCLIB do not have corresponding members in PROD.LOADLIB

3. Which members of PROD.LOADLIB do not have corresponding members in PROD.SOURCLIB.

The first step is to execute the DUMPDIR macro for PROC.SOURCLIB and then for PROC.LOADLIB. The DUMPDIR macro performs the following tasks:

1. Allocate the library (PDS) that is to have its directory processed.
2. Allocate a temporary flat file to accept the directory records from PROC SOURCE.
3. Dump the library directory to the flat file via PROC SOURCE. The directory records are dumped to the flatfile by way of the DIRDD option of the PROC SOURCE statement.
4. Use a DATA Step to read the flat file and create a SAS data set with observations that contain the name of each member of the library. (Remember that "member name" is synonymous with "program name").
5. Sort the new SAS data set by member name.
6. Deallocate all files.

In **Example 1**, the DUMPDIR macro is executed first with PROD.SOURCLIB and then with PROD.LOADLIB as input. This creates SAS data sets PERM.SOURCLIB and PERM.LOADLIB, respectively. Then, a match merge is performed with the two data sets to create three new SAS data sets:

PERM.MATCHES - Contains members of PROD.SOURCLIB that match members of PROD.LOADLIB.

PERM.ONLYSOUR - Contains members of PROD.SOURCLIB that have no matches with members of PROD.LOADLIB.

PERM.ONLYLOAD - Contains members of PROD.LOADLIB that have no matches with members of PROD.SOURCLIB.

Example 1 effectively produces an inventory of source library and load library matches and non-matches. Armed with this information, programmers can determine how many source/load COBOL programs must be examined for Year 2000 compliance. Programmers can discover which load library programs do not have source code in the (production) source library. And, programmers can learn which source programs have never been compiled and linked into the load library.

Determining Which Programs Are Executed

Another common task in a Year 2000 compliance project is to determine which production batch COBOL programs are currently being executed. Since COBOL batch programs are executed via JCL, the production JCL library must be processed. But, unlike **Example 1** where the library member names were scrutinized, the actual contents of each library member must be examined. Only by probing every line of every member of the production JCL library can the program names be identified. PROC SOURCE is ideally suited to this task.

Example 2 exploits PROC SOURCE's ability to create a flat file containing records for the IBM IEBUPDTE utility. You can create this specific type of output by using the OUTDD option in the PROC SOURCE statement. The flat file that PROC SOURCE produces contains the contents of every member of the production JCL library placed one-after-another in alphabetical order. The contents of each production JCL library member is preceded with a single record that contains the following, starting in column 1:

```
./     ADD   NAME=xxxxxxxx
```

In the form, above, xxxxxxxx is the 1 to 8 character production JCL library member name. It is there to specify that the following records are from the production JCL library member with that particular member name. Here is an example of a portion of an IEBUPDTE-type flat file:

```
./     ADD   NAME=BIGJOB01
//PRODJOB1 JOB (ACCT1,ACCT2) 'BIG PRODJOB',
//   MSGLEVEL=(1,1),CLASS=A,MSGCLASS=H
//STEP01 EXEC PGM=PROD0001
//INLIB  DD DSN=PROD.INDATA.FLATFILE,DISP=SHR
//OUTLIB DD DSN=PROD.OUTDATA.FLATFILE,DISP=OLD
//*
//STEP02 EXEC BIGPROG2
//INPUT  DD DSN=PROD.GOODATA.FLATFILE,DISP=SHR
//OUTPUT DD DSN=PROD.BADATA.FLATFILE,DISP=OLD
//
./     ADD   NAME=BIGJOB02
//PRODJOB2 JOB(ACCT1,ACCT2) 'ANOTHER PRODJOB',
. . . . . etc. . . . . .
```

The SAS code in **Example 2** performs the following tasks:

1. Allocate the production JCL library (PDS) that is to be processed.
2. Allocate a temporary flat file to store the IEBUPDTE utility records output from PROC SOURCE.
3. Dump the production JCL library to the flat file via PROC SOURCE. The records are dumped to the flatfile by way of the OUTDD option of the PROC SOURCE statement.
4. Use a DATA Step to read the flat file and select only records that contain either:
   1. "./" in the first 2 columns and "ADD" in column 10.
   2. "//" in the first 2 columns, "EXEC" in a subsequent column, and DO NOT contain a '*' in column 3. These records probably execute

programs, and thus contain a Program Name.

5. Read the SAS data set to create observations that contain the Job Name (from the "ADD" records) along with each Program Name. Delete observations that overtly contain 'PROC=", since they invoke PROCs, not programs.
6. Make another pass of the SAS data set to delete observations that contain programs that are known to be either: IBM utilities, third-party vendor proprietary programs, or in-house PROCs. This is done via the user-created $UTILITY SAS Format.

The final output of **Example 2** is a SAS data set that contains all programs that are currently in the production JCL library. The label of the SAS data set was assigned to be the production JCL library name, for future identification. Each observation in the data set contains the member name (job name) and a program name.

This information is valuable for a number of reasons. It can be used to determine the number of batch programs that may need conversion to Year 2000 compliance. The output can be matched against the load library output of **Example 1** to determine if there are programs in the load library that are no longer being executed in production. The information can be used to determine if production programs are running from libraries other than the production load library. These are only a few of the many uses that the output from **Example 2** can be used for.

Search Source Programs for Year Variables

Example 3 uses PROC SOURCE to search through a source library to find instances of known year variables. In this example, good programming standards have compelled an organization's programmers to use a handful of variable names to represent year values. Consequently, determining which programs may need date field conversions is fairly straight-forward. It is simply a matter of searching through the programs in the source library for those that contain occurances of one of the year variables. These will be the programs that need further attention from the Year 2000 conversion team.

Example 3 completes the following tasks:

1. Allocate the source library (PDS) that is to be processed.
2. Allocate a temporary flat file to store the IEBUPDTE utility records output from PROC SOURCE.
3. Dump the source library to the flat file via PROC SOURCE. The records are dumped to the flatfile by way of the OUTDD option of the PROC SOURCE statement.
4. Use a DATA step to process the flat file. Each line is scanned to determine if it contains one of the Year variables that are stored in the user-created YEAR SAS Format. If a line does contain a Year variable, a SAS observation is created. That observation contains:

   1. The library member name
   2. The Year variable name
   3. The line number where the variable was found

The SAS data set created in item #4, above, is the final output of **Example 3**. By using the information in this SAS data set, programmers can be selective about the programs that they scrutinize for Year 2000 date changes. They can zero in on specific programs, and on the lines that need to be examined for possible modifications. This can be a great help in an organization that has formalized variable naming standards.

Conclusion

The examples presented in this paper illustrate three ways that PROC SOURCE can be used to ferret out Year 2000 compliance problems. The bedrock of each example is PROC SOURCE's ability to methodically process either the directory of a PDS or the contents of each member of a PDS. By coupling PROC SOURCE with SAS character handling functions, you have a powerful set of tools to use in your own organization's Year 2000 conversion project. So, what are you waiting for? Get started today!

Trademarks

References

SAS Institute Inc
SAS Procedures Guide, Version 6, Third Edition
Cary, NC:SAS Institute Inc., 1990
705pp.

Raithel, Michael A.
Tuning SAS Applications in the MVS Environment
Cary, NC:SAS Institute Inc., 1995
303pp.

Contact Information

Michael A. Raithel
PO Box 406
Garrett Park, Md. 20896
E-mail: maraithel@mcimail.com

EXAMPLE 1:

Matching Source Programs with Load Modules

```
*******************************************************;
* ALLOCATE THE SAS DATA LIBRARY THAT WILL BE *;
* USED TO STORE THE OUTPUT DATA SETS.              *;
*******************************************************;
LIBNAME PERM  'YOUR.BIG.SASLIB' DISP=OLD;


 /*******************************************************/
 /* DUMPDIR MACRO TO DISSECT THE PDS DIRECTORY.*/
 /*******************************************************/
%MACRO DUMPDIR(PDSNAME,SASFILE);

*******************************************************;
* ALLOCATE THE PDS THAT WILL HAVE ITS DIRECTORY  *;
* PROCESSED.                                       *;
*******************************************************;
FILENAME PDS "&PDSNAME" DISP=SHR;

*******************************************************;
* CREATE A TEMPORARY DATA SET TO HOLD THE OUTPUT*;
* DIRECTORY RECORDS OF 'PROC SOURCE'.                *;
*******************************************************;
FILENAME TEMP '&TEMPLIB'
       DISP=NEW
       SPACE=(CYL,(1,1)) UNIT=SYSDA
       RECFM=FB LRECL=80
       ;

*******************************************************;
* USE 'PROC SOURCE' TO DUMP THE TARGET PDS          * ;
* DIRECTORY TO THE  'TEMP' FLAT FILE.               *;
*******************************************************;
PROC SOURCE INDD=PDS DIRDD=TEMP NODATA NOPRINT;
RUN;

*******************************************************;
* INPUT THE MEMBER NAMES FROM THE 'TEMP' FLAT FILE*;
*******************************************************;
DATA PERM.&SASFILE(LABEL="&PDSNAME");

INFILE TEMP;

LABEL PROGNAME = 'PROGRAM*NAME';

INPUT @1  PROGNAME   $CHAR8.;

RUN;

*******************************************************;
* SORT BY PROGRAM NAME.                           *;
*******************************************************;
PROC SORT;
   BY PROGNAME;
RUN;

*******************************************************;
* CLEAR THE FILENAMES FOR THE NEXT MACRO EXECUTION*;
*******************************************************;
FILENAME PDS CLEAR;

FILENAME TEMP CLEAR;

 /*******************************************************/
 /* END DUMPDIR MACRO.                                  */
 /*******************************************************/
%MEND DUMPDIR;


 /*******************************************************/
 /* INVOKE DUMPDIR MACRO.                               */
 /*******************************************************/
%DUMPDIR(PROD.SOURCLIB,SOURCLIB);
%DUMPDIR(PROD.LOADLIB,LOADLIB);


*******************************************************;
* MATCH THE SOURCE AND LOAD LIB MEMBERS           *;
*******************************************************;
DATA PERM.MATCHES
         PERM.ONLYSOUR
         PERM.ONLYLOAD;
MERGE PERM.SOURCLIB(IN=A)
          PERM.LOADLIB(IN=B);
            BY PROGNAME;
            IF A AND B THEN OUTPUT PERM.MATCHES;
            ELSE IF A AND NOT B THEN OUTPUT PERM.ONLYSOUR;
```

```
            ELSE IF B AND NOT A THEN OUTPUT PERM.ONLYLOAD;
RUN;
```

EXAMPLE 2:

Determining Which Programs Are Executed

```
*************************************;
* KNOWN PROCS AND IBM UTILITIES *;
*************************************;
PROC FORMAT;
   VALUE $UTILITY
        'HEWL    '      =            'Y'
        'IDCAMS  '      =            'Y'
        'IEBCOPY '      =            'Y'
        'IEBEDIT '      =            'Y'
        'IEBGENER'      =            'Y'
        'IEBPTPCH'      =            'Y'
        'IEFBR14 '      =            'Y'
        'IEHPROGM'      =            'Y'
        'IEV90   '      =            'Y'
        'IKJEFT01'      =            'Y'
        'SAS     '      =            'Y'
         OTHER          =            'N'
         ;

 /*****************************************************/
 /* ALLOCATE THE SAS DATA LIBRARY WHERE THE */
 /* OUTPUT DATA SETS WILL BE STORED.                  */
 /*****************************************************/
LIBNAME PDSFILE  'YOUR.PROD.SASLIB'  DISP=OLD;


 /***********************************************/
 /* MACRO TO PROCESS PDS MEMBERS            */
 /***********************************************/
%MACRO PDSMEMBS(PDSNAME,SASFILE);

************************************************;
* ALLOCATE THE PDS THAT WILL HAVE ITS         *;
* MEMBERS PROCESSED.                          *;
************************************************;
FILENAME PDS "&PDSNAME" DISP=SHR;

************************************************;
* CREATE A TEMPORARY DATA SET TO HOLD THE *;
* OUTPUT CONTENTS OF MEMBERS IN THE PDS      *;
*PROCESSED BY  'PROC SOURCE'.                 *;
************************************************;
FILENAME MEMBERS '&TEMPFILE'
       DISP=NEW
       SPACE=(CYL,(50,10)) UNIT=SYSDA
       RECFM=FB LRECL=80;

************************************************;
* USE PROC SOURCE TO DUMP THE TARGET PDS  *;
* TO THE 'MEMBERS' FLAT FILE.             *;
************************************************;
PROC SOURCE INDD=PDS OUTDD=MEMBERS NOPRINT;
RUN;

************************************************************;
* INPUT RECORDS IN THE 'MEMBERS' FLAT FILE, PROCESS *;
* THEM AND ONLY KEEP LINES CONATINING EITHER THE *;
* MEMBER NAME OR AN EXEC STATEMENT.                  *;
************************************************************;
DATA PDSFILE.RAWRECS;

INFILE MEMBERS;

INPUT @1  STRING   $CHAR80.;

IF (STRING =: './' AND SUBSTR(STRING,10,3) = 'ADD') OR
  (SUBSTR(STRING,1,2) = '//' AND SUBSTR(STRING,3,1) NE '*' AND
  INDEX(STRING,'EXEC') > 0) THEN
      OUTPUT;

RUN;


************************************************************;
* SECOND PASS OF THE DATA SET CREATED IN THE              *;
* PREVIOUS DATA STEP. THIS TIME, GET THE MEMBER           *;
* NAMES AND THE SUBSEQUENT 'EXEC' CARDS ANDWRITE *;
* THEM OUT TO THE OUTPUT SAS DATA SET.                    *;
************************************************************;
DATA PDSFILE.&SASFILE(KEEP=JOBNAME PROGNAME
                          LABEL="&PDSNAME");
SET PDSFILE.RAWRECS END=EOF;

RETAIN JOBNAME ;

LABEL JOBNAME    = 'JOB*NAME'
          PROGNAME = 'PROGRAM*NAME';

LENGTH PROGNAME $8.;

SELECT(SUBSTR(STRING,1,2));

WHEN('./') DO;

  X = (INDEX(STRING,'NAME=')) + 5;
  JOBNAME = SUBSTR(STRING,X,8);
  DELETE;

END;

WHEN('//') DO;

  IF INDEX(STRING,'PROC=') > 0 THEN DELETE;

  BEGINPGM = (INDEX(STRING,'PGM=') + 4);

  IF BEGINPGM > 4 THEN DO;

ENDPGM = INDEXC(SUBSTR(STRING,BEGINPGM,(80-BEGINPGM)),' ',',')
-1;
    PROGNAME = SUBSTR(STRING,BEGINPGM,ENDPGM);
    OUTPUT;

  END;

  ELSE DO;
    ENDEXEC  = (INDEX(STRING,'EXEC') + 4);
    STARTPGM =
           INDEXC(SUBSTR(STRING,ENDEXEC,(80-ENDEXEC)),
           'ABCDEFGHIJKLMNOPQRSTUVWXYZ') + ENDEXEC -1;

    ENDPGM =
        INDEXC(SUBSTR(STRING,STARTPGM,(80-STARTPGM)),' ',',') -1;

    PROGNAME = SUBSTR(STRING,STARTPGM,ENDPGM);

    OUTPUT;

   END;

END;

  OTHERWISE;

END;

RUN;
```

```
*************************************************************;
* THIS STEP DELETES JOBS WITH KNOWN PROCS OR IBM *;
* UTILITIES.                                                 *;
*************************************************************;
* NOTE:  YES, YES, YES, THIS COULD HAVE EASILY BEEN *;
* DONE, ABOVE, BUT, I PUT IT HERE TO PRESERVE THE    *;
* CLARITY (AND BEAUTY) OF THE CODE.                  *;
*************************************************************;
DATA PDSFILE.&SASFILE;
SET     PDSFILE.&SASFILE;

IF PUT(PROGNAME,$UTILITY.) = 'Y' THEN DELETE;

RUN;

 /*********************************************/
 /* END MACRO TO PROCESS PDS MEMBERS     */
 /*********************************************/
%MEND PDSMEMBS;

 /*********************************************/
 /* INVOKE MACRO TO PROCESS PDS MEMBERS */
 /*********************************************/
%PDSMEMBS(PROD.JCLLIB,PRODPROG)
```

EXAMPLE 3:

Search Source Programs For Year Variables

```
*********************************;
* KNOWN YEAR VARIABLES         *;
*********************************;
PROC FORMAT;
    VALUE YRVARS
         1  = 'YEAR'
         2  = 'FISCAL-YEAR '
         3  = 'YEAR-MONTH-DAY '
         4  = 'CALENDAR-YEAR '
         5  = 'SALESYEAR'
         ;

  /*****************************************************/
  /* ALLOCATE THE SAS DATA LIBRARY WHERE THE  */
  /* FINAL DATA SETS WILL BE STORED.                   */
  /*****************************************************/
LIBNAME PDSFILE 'YOUR.PROD.SASLIB'  DISP=OLD;

 /*********************************************/
 /* MACRO TO PROCESS PDS MEMBERS           */
 /*********************************************/
%MACRO PDSMEMBS(PDSNAME,SASFILE);

*************************************************;
* ALLOCATE THE PDS THAT WILL HAVE ITS     *;
* MEMBERS PROCESSED.                      *;
*************************************************;
FILENAME PDS "&PDSNAME" DISP=SHR;

*************************************************;
* CREATE A TEMPORARY DATA SET TO HOLD THE *;
* OUTPUT CONTENTS OF MEMBERS IN THE PDS       *;
* PROCESSED BY 'PROC SOURCE'.                 *;
*************************************************;
FILENAME MEMBERS '&TEMPFILE'
        DISP=NEW
        SPACE=(CYL,(50,10)) UNIT=SYSDA
        RECFM=FB LRECL=80;

*************************************************;
* USE PROC SOURCE TO DUMP THE TARGET PDS TO *;
* THE 'MEMBERS' FLAT FILE.                  *;
*************************************************;
PROC SOURCE INDD=PDS OUTDD=MEMBERS NOPRINT;
RUN;

*************************************************************;
* INPUT THE RECORDS IN THE 'MEMBERS' FLAT FILE, PROCESS *;
* THEM AND ONLY KEEP LINES THAT REFERENCE YEAR          *;
* VARIABLES.                                            *;
*************************************************************;
DATA PDSFILE.&SASFILE(KEEP=MEMNAME YEARVAR LINENUM
                LABEL="&PDSNAME");

LENGTH YEARVAR MEMNAME $8.;

RETAIN MEMNAME LINENUM;

INFILE MEMBERS;

INPUT @1  STRING   $CHAR80.;

IF (STRING =: './' AND SUBSTR(STRING,10,3) = 'ADD') THEN DO;
   X = (INDEX(STRING,'NAME=')) + 5;
   MEMNAME = SUBSTR(STRING,X,8);
    LINENUM = 0;
    YEARVAR = '       ';
   DELETE;
END;
ELSE DO;
    LINENUM + 1;

    DO I = 1 TO 5;
       IF INDEX(STRING,(TRIM(PUT(I,YRVARS.)))) > 0 THEN DO;
          YEARVAR = TRIM(PUT(I,YRVARS.));
          OUTPUT;
       END;
    END;
END;
RUN;

 /*********************************************/
 /* END MACRO TO PROCESS PDS MEMBERS      */
 /*********************************************/
%MEND YEARVARS;

 /*********************************************/
 /* INVOKE MACRO TO PROCESS PDS MEMBERS */
 /*********************************************/
%YEARVARS(PROD.COBOL.SOURCLIB,PRODSOUR);
```