

Utilizing VMS Directory Information Through SAS® Finding SAS Datasets and Catalogs Without Knowing Where to Look

Shari Forvendel
Pharmaceutical Product Development, Inc.,
Morrisville, NC

ABSTRACT

As the Biostatistical/Data Management division of a Contract Research Organization, my company maintains libraries of data for numerous projects in an OpenVMS/AXP environment. The data for each project are located in a directory tree and defined by a unique system logical. Within this structure it is possible to use the SAS system to access specific project information using the X statement and VMS/DCL commands.

This paper will demonstrate an application using the X statement to locate project files within the directory structure and pass that information to LIBNAME statements in a SAS program. This technique, combined with the SAS macro language, can make your utility programming generic enough to be used with any project without requiring major modifications for each use.

BACKGROUND

During a recent system upgrade, it was necessary to convert all SAS datasets and catalogs for several complex projects to transport files to be ported from the existing VAX/VMS environment to the new ALPHA/OpenVMS platform. Each project was stored in a separate hierarchical directory structure that could be referenced by a unique system logical (see definition below).

The difficulty was not in the porting process, but in locating all the datasets and catalogs without having to do a search through all the directories and then convert each file independently. The program outlined in this paper was created to perform both the search process and the conversion to transport files. It also has the added feature of handling multiple projects with one execution.

DEFINITIONS

System Logical	A string of characters that is usually equated to a file name, device name, or other logical name. ¹ For this application the project name is the logical defining the directory tree for the project data. For example: PROJECT_NAME:[DATA] defines the data directory. This allows the project to be referenced with a "generic" term without having to know the exact disk location.
Protocol	For this application, <i>protocol</i> is synonymous with <i>project</i> . Each protocol is an individual contracted study.
PURGE command	VAX/VMS command that deletes all except the latest version of all files in the specified directory(s).
Engine	Sets of internal instructions the SAS system uses to read from and write to files. Engines open files, direct input/output operations, and gather descriptive information about files and their contents. ²

APPLICATION

```

*****
/* PROGRAM: Archive.SAS
/* AUTHOR: Shari H. Forvendel
/* CREATED: 02/27/95
/* PURPOSE: Locate all SAS datasets and catalogs
/*           for a given project or projects and
/*           create transport files for each of them.
/* USAGE: Modify project name list in Part I and
/*         run program through batch process.
/*
/* MODIFICATION LOG:
/*
/* Copyright 1995, PPD, Inc.
/* All Rights Reserved.
*****

*-----*
* PART I: Project(s) list
*-----*

%let proj1=protocol_name1;
%let proj2=protocol_name2;

%let totproj=2;

*-----*
* create a work area to contain all temporary files.
*-----*

%let listdir=disk5:[sastemp.archive];
X create/dir &listdir;

```

Part I of the program provides a list, in the form of macro variables, of the projects to be processed and a counter variable to indicate the number of projects processed. This list can be generated in a number of ways, specifying any number of projects, including reading in an external file and using CALL SYMPUT statements to assign the values to the macro variables. For this example, however, %LET statements were used to assign two protocol names and the project counter to the macro variables PROJ1, PROJ2, and TOTPROJ, respectively. The counter variable TOTPROJ will be used to control a %DO loop to cycle through each of the projects. This is the only section that may need to be modified prior to each run of the program.

Lastly, Part I issues an X statement to create a work area to write files created by this program. Creating permanent files, as opposed to utilizing SAS's WORKLIB, allows for ease in tracing the information being used by the program should anything appear to go wrong during execution.

Figure 1: Work file from DIR command

```

*-----*
* PART II: Cycle through projects: locate SAS files and
*          convert them to transport files.
*-----*

%MACRO TRN_PROJ;

  %DO I=1 %to &totproj;

    *-----*
    * start log for each project on a new page
    * with a header to identify the current project
    *-----*

    page;
    %put .....;
    %put * Creating transport files for &&proj&i .....;
    %put .....;

    *-----*
    * purge project directories to save space.
    *-----*

    X purge &&proj&i...[000000...];

    *-----*
    * create a list of all directories for the project
    * that contain SAS datasets or SAS catalogs.
    *-----*

    %let filelist=&listdir.&&proj&i...list;

    X dir/col=1/out=&filelist &&proj&i...[000000...]*.sas*$* ;
    run;

```

Part II of the program begins the definition of the macro that processes the projects. The %DO loop cycles through the list of projects created in Part I.

The search through the project area for all SAS datasets and catalogs is performed by X statements to issue system commands. The first X statement executes the PURGE command to "clean up" the project area to avoid reporting on duplicate files. The syntax [000000...] indicates that the command begins at the root directory of the given project and continues through all the nested directories.

It is the DIR command that actually serves as the search procedure. The DIR command creates a list of all files in the project area that contain the nomenclature of a SAS dataset or catalog and outputs the list to an external file created by the OUT= option. All SAS catalogs are stored on the VMS system as XXX.saseb\$catalog and all datasets (not currently being accessed) are saved as XXX.sasiorms\$data or XXX.saseb\$data, depending on the engine specified when they were created. Therefore, the text in the search string is *.sas*\$*, where * is a wild card.

The result of the DIR command is a file containing file names for only those directories containing SAS catalogs and datasets. (Figure 1). The work file name is the project name (example: *project_name1.list*) as assigned by the preceding %LET statement. The COL=1 option on the DIR command lists each file or directory on a separate line. This will enable them to be read in as individual records with minimal effort.

```

Directory protocol_name1:[000000.COMPARE.950306S_B.ENTRY1]

FINAL.SASIORMS$DATA;1
LAB.SASIORMS$DATA;1
MEF.SASIORMS$DATA;1
NIM.SASIORMS$DATA;1
SELECT.SASIORMS$DATA;1

Total of 5 files.

Directory protocol_name1:[000000.COMPARE.950306S_B.ENTRY2]

FINAL.SASIORMS$DATA;1
LAB.SASIORMS$DATA;1
MEF.SASIORMS$DATA;1
NIM.SASIORMS$DATA;1
SELECT.SASIORMS$DATA;1

Total of 5 files.

Directory protocol_name1:[000000.DATA.DM]

FINAL.SASIORMS$DATA;22
FOLLOW.SASIORMS$DATA;26
LAB.SASIORMS$DATA;44
MEF.SASIORMS$DATA;45
NIM.SASIORMS$DATA;44
SELECT.SASIORMS$DATA;44

Total of 6 files.

Directory protocol_name1:[000000.FORMAT]

FORMATS.SASEB$CATALOG;9

Total of 1 file.

Directory protocol_name1:[000000.APPL]

DB_INDEX.SASIORMS$DATA;7
SCREEN.SASEB$CATALOG;6

Total of 2 files.

Directory protocol_name1:[000000.PROGRAM.MAINTDM.DATA]

FORMATS.SASIORMS$DATA;5
FORMVAL.SASIORMS$DATA;5
MAKE_DS.SASIORMS$DATA;35
VALUES.SASIORMS$DATA;5

Total of 4 files.

Grand total of 6 directories, 23 files.

```

- PART III: Read the directory list and create a dataset with directory name and determine whether the directory contains SAS catalogs and/or SAS datasets. (Also, determine the engine used for datasets).

```
FILENAME list "&filelist";
```

```
DATA FILES (keep=protcl dir category);
```

```
length protcl dir category $200 test $10;
```

```
retain protcl "&proj&" dir "";
```

```
infile list missover irec=200 pad end=eof;
```

```
input @1 line $100.;
```

```
if (line ne "");
```

```
recnt+1;
```

```
test=substr(left(line),1,9);
```

```
if (test="Directory") then DIR=scan(line,2," ");
```

```
else do;
```

```
  if (index(line,"$")) then do;
```

```
    if index(upcase(line),"SASEB$CATALOG")
```

```
      then category="catalog";
```

```
    else
```

```
    if index(upcase(line),"SASIORMS$DATA")
```

```
      then category="concur";
```

```
    else
```

```
    if index(upcase(line),"SASEB$DATA")
```

```
      then category="dataset";
```

```
    else put @2 "WARNING: The following"
```

```
      "file does not conform to "
```

```
      "the program categories."
```

```
      / @2 "No transport file was created."
```

```
      // @6 "Directory: " dir ;
```

```
  OUTPUT;
```

```
end;
```

```
end;
```

```
RUN;
```

```
PROC SORT data=files NODUPKEY;
```

```
  by protcl dir category;
```

```
run;
```

The ultimate purpose of this program is to create transport files for all datasets and catalogs within a directory. Because SAS can do this for an entire library in one command, it is not necessary to know the actual dataset or catalog name, only its type (catalog or dataset) to determine the method needed to convert it to a transport file. For datasets, the engine (concur or V607) is also needed to apply to any LIBNAME statements used during the conversion.

Part III of the program reads the external file created in Part II to create a dataset with all the necessary file information. Omitting all blank lines, each record is evaluated using the SUBSTR or INDEX function to determine the type of information it contains: either a directory name, a dataset name, or a catalog name.

Each directory name is assigned to the variable DIR and retained for each file to preserve its location.

The file extensions for datasets and catalogs are used to determine the type (catalog or dataset) or engine (concur or V607). This information is assigned to the variable CATEGORY based on the criteria outlined in Figure 2.

The protocol name is retained in the variable PROTCL for debugging and verification purposes only.

Figure 2: CATEGORY criteria

CATEGORY	Criteria
catalog	SAS catalog (format, application, etc.)
concur	SAS dataset created using the CONCUR engine.
dataset	All other SAS datasets.

Once the dataset is created, it is sorted using the NODUPKEY option. The resulting dataset contains one observation per category for each directory. (see Figure 3)

Figure 3: PROC PRINT of file created by DIR command

The SAS System		09:57 Wednesday, June 28, 1995
OBS	PROTCL	
1	project_name1	
2	project_name1	
3	project_name1	
4	project_name1	
5	project_name1	
6	project_name1	
OBS	DIR	
1	project_name1:[000000.COMPARE.950306S_B.ENTRY1]	
2	project_name1:[000000.COMPARE.950306S_B.ENTRY2]	
3	project_name1:[000000.DATA.DM]	
4	project_name1:[000000.FORMAT]	
5	project_name1:[000000.APPL]	
6	project_name1:[000000.APPL]	
7	project_name1:[000000.PROGRAM.MAINTDM.DATA]	
OBS	CATEGORY	
1	concur	
2	concur	
3	concur	
4	catalog	
5	catalog	
6	concur	
7	concur	
[End of file]		

- PART IV: Create macro variables from directory information. Loop through directories and create transport files using the appropriate procedure.

```

DATA _NULL_;
  SET FILES end=eof;
  call symput('dir' || left(put(_n_4.)), trim(dir));
  call symput('cat' || left(put(_n_4.)), trim(category));
  if (eof) then
    call symput('totdirs', left(put(_n_4.)));
run;

%DO D=1 %TO &TOTDIRS;

  %put - Creating transport file for directory &dir&d ;
  %put - Category = &cat&d ;

  %IF ( %upcase(&cat&d) = CONCUR ) %then
    %do;
      LIBNAME concurs concur "&dir&d";
      LIBNAME tranfile xport "&dir&d..concur.tm";
      PROC COPY in=concurs
                out=tranfile
                memtype=data;

      run;
    %end;

  %IF ( %upcase(&cat&d) = DATASET ) %then
    %do;
      LIBNAME data "&dir&d";
      LIBNAME tranfile xport "&dir&d..data.tm";
      PROC COPY in=data
                out=tranfile
                memtype=data;

      run;
    %end;

  %IF ( %upcase(&cat&d) = CATALOG ) %then
    %do;
      LIBNAME catdir "&dir&d";
      FILENAME tranfile "&dir&d..catalogs.tm";
      PROC CPORT library=catdir
                memtype=catalog
                file=tranfile;

      run;
    %end;

%END; /* end loop through directories */

%END; /* end loop through projects */

%MEND ; /* end macro TRN_PROJ */

%TRN_PROJ;

```

The final portion of the program, Part IV, utilizes the information from the dataset created in Part III to perform the appropriate porting procedure.

The first step in doing this is to convert the directory names and categories to macro variables using the CALL SYMPUT statement within a data step. The macro variables for each record are numbered sequentially using the automatic data step variable _N_ (see figure 4). As in Part I, a counter variable is constructed to be used in a %DO loop. For this example, macro variable TOTDIRS equals 7, the number of records in the FILES dataset.

Figure 4: Macro variables with directory and file type information

Var	Macro Variable Value
&DIR1	project_name1:[000000.COMPARE.950306S_B.ENTRY1]
&CAT1	concur
.	.
.	.
&DIR4	project_name1:[000000.FORMAT]
&CAT4	catalog
.	.
.	.
&DIR7	project_name1:[000000.PROGRAM.MAINTDM.DATA]
&CAT7	concur

The final step is to cycle through the directory list. The %DO loop executes the appropriate SAS porting procedure based on the value of the current category macro variable. Since the values 'CONCUR' and 'DATASET' both refer to SAS datasets a PROC COPY is performed in both instances. The only difference between them is the engine specified on the LIBNAME statement.

For this application, the transport files are written to the same directory where the original files are located using the DIRx macro variables. This was a deliberate design to aid in the importing process. For importing the files to the OpenVMS operating system, a similar program was written to locate the files CONCUR.TRN, DATA.TRN, and CATALOGS.TRN and perform the necessary import procedure.

CONCLUSION

The program presented here turns a painstaking task into a fully automated process. By using the SAS X statement to provide host directory information to the program, it is not necessary to know ahead of time the location or type of the target files.

The use of the SAS macro language provides the mechanism to execute the appropriate SAS code based on the directory information. Also, by using it to construct the project list, the program is generic enough to be used repeatedly with minimal modification.

REFERENCES

- ¹SAS Institute Inc., *SAS® Language: Reference, Version 6, First Edition* Cary, NC: SAS Institute Inc., 1990. 1042 pp.
- ²*VMS User's Manual*, Maynard, Massachusetts: Digital Equipment Corporation, 1989

ACKNOWLEDGMENTS

This program presented in this paper was developed using Release 6.07 of the SAS System on a Digital Equipment Corporation VAX™ machine using version 5.5-2 of the VMS™ operating system.

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries. VAX and VMS are trademarks of Digital Equipment Corporation. ® denotes USA registration.

The author acknowledges the input, suggestions, and cooperation of the Clinical Computing Department of Pharmaceutical Product Development, Inc. in the completion of this work.

FOR FURTHER INFORMATION

The author can be reached as listed below:

Shari H. Forvendel
Pharmaceutical Product Development, Inc.
1400 Perimeter Park Drive, Suite 100
Morrisville, NC 27560
Telephone: (919) 380-2000
FAX: (919) 380-2022
Internet: forvendelsh%ppdrtp@mcimail.com

Pharmaceutical Product Development, Inc. is a full-service clinical Contract Research Organization.