Paper 3458-2015

Better Metadata Through SAS® II: %SYSFUNC, PROC DATASETS, and Dictionary Tables

Louise Hadden, Abt Associates Inc., Cambridge, MA

ABSTRACT

SAS® provides a wealth of resources for users to create useful, attractive metadata tables, including PROC CONTENTS listing output (to ODS destinations), the PROC CONTENTS OUT= SAS data set, and PROC CONTENTS ODS Output Objects. This paper and presentation explore some less well-known resources to create metadata such as %SYSFUNC, PROC DATASETS, and Dictionary Tables. All these options in conjunction with the use of the EXCELXP (and new in 9.4 M2, the EXCEL) tagsets enable the creation of multi-tab metadata workbooks at the click of a mouse.

INTRODUCTION

In a prior paper (Hadden, 2014) I discuss using PROC CONTENTS, ODS OUTPUT and the EXCELXP tagset to create multi-tab documentation (metadata) spreadsheets to describe data sets. This paper and presentation build on that concept, exploring several alternative procedures and tools to view and create SAS metadata. Although all the tools discussed have utility beyond documentation purposes, other uses are beyond the scope of this paper. Users are encouraged to read the papers referred to in the References section for more information, and to explore these great resources on their own.

Who cares about metadata? Any SAS programmer should care! Knowing your data is vital. By using SAS metadata in conjunction with careful documentation practices that some of the tools we are discussing make easy, you can find out when a program was last run, who ran it, what variables were created, whether the data set is sorted or indexed, and more. Follow me down the metadata worm hole and we will find out how….

RESOURCES WITHIN SAS® TO OBTAIN METADATA

Every time a SAS session is initiated, a wealth of metadata becomes available to users. The SAS system collects information on libraries and data sets specified in LIBNAME and FILENAME statements, libraries and data sets automatically available to SAS users, data sets and files created within the session, system and version information, host specific information such as usernames, among others. This paper discusses five methods of accessing this information (PROC CONTENTS having been covered earlier, making six methods.)

1) PROC CONTENTS (not covered here)
2) PROC DATASETS
3) SAS Dictionary Tables (used with PROC SQL)
4) SASHELP Views (can be used with PROC SQL and with the SAS data step)
5) %SYSFUNC (access to metadata through MACRO functions)
6) “V” functions (access to metadata through data set functions)

I will discuss resources 2 through 6, briefly describing the resource in relation to metadata, and showing an example of using each resource. I hope the utility of each resource, whichever one(s) appeal to a given SAS programmer, will be obvious.

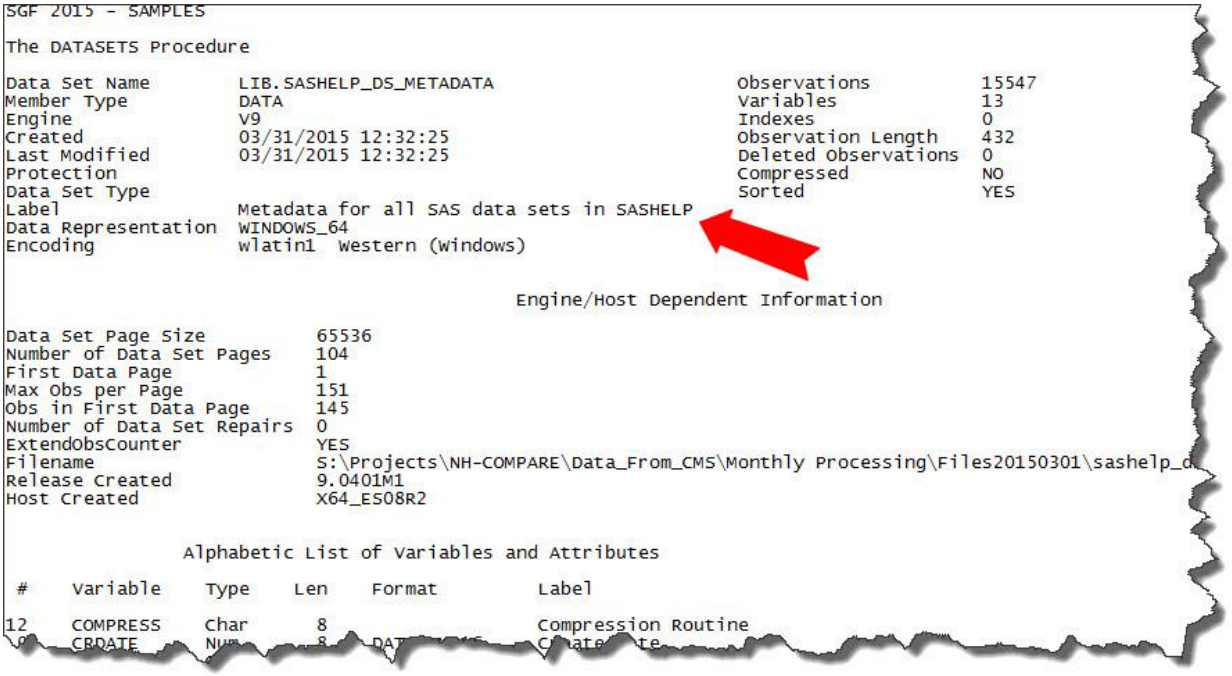
RESOURCE #2. PROC DATASETS

PROC DATASETS does all that PROC CONTENTS does, and much, much more. The primary use I will discuss is the CONTENTS listing, although we will touch on PROC DATASETS ability to modify datasets by adding labels, formats, etc. An advantage that PROC DATASETS has over PROC CONTENTS is that it can produce a single file containing documentation on multiple data sets in a library with a single command.

```
* RESOURCE 2: PROC DATASETS;

/* generate a contents listing of all data sets in SASHELP */
 PROC DATASETS LIBRARY=SASHELP NOLIST;
```

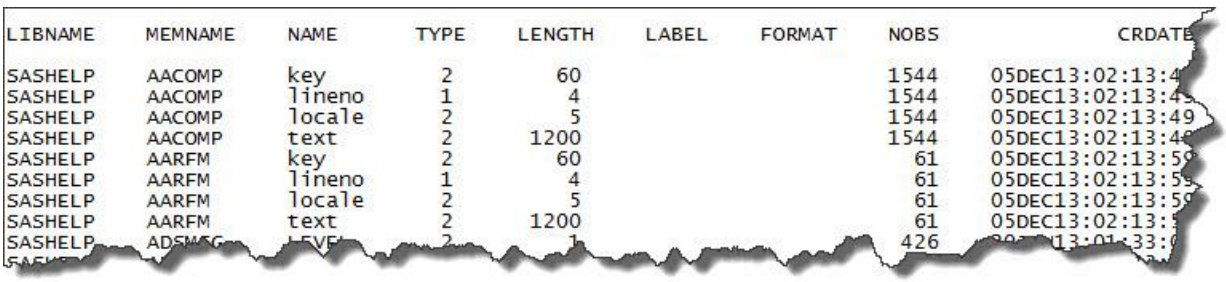
```
    CONTENTS DATA=_ALL_
    OUT=LIB.SASHELP_DS_METADATA (KEEP=LIBNAME MEMNAME NAME TYPE LENGTH LABEL
   FORMAT NOBS COMPRESS SORTED IDXUSAGE CRDATE MODATE)
    NOPRINT DIRECTORY;
RUN;

/* Add a data set label */
PROC DATASETS LIBRARY=LIB;
    MODIFY SASHELP_DS_METADATA
    (LABEL="Metadata for all SAS data sets in SASHELP");
    CONTENTS DATA=SASHELP_DS_METADATA;
RUN:
QUIT;
```

A contents listing and a test print of the resulting data set (LIB.SASHELP_DS_METADATA) is listed in ***Outputs 1 and 2*** *respectively.* Note that some variables in the output data set are on the data set level, while others are on the variable level, just as in PROC CONTENTS. ODS OUTPUT objects from PROC DATASETS output can be manipulated just as they can be for PROC CONTENTS, and the output data set can be manipulated with data step(s) as well.



```
SGF 2015 - SAMPLES

The DATASETS Procedure

Data Set Name        LIB.SASHELP_DS_METADATA                 Observations          15547
Member Type          DATA                                    Variables             13
Engine               V9                                      Indexes               0
Created              03/31/2015 12:32:25                     Observation Length    432
Last Modified        03/31/2015 12:32:25                     Deleted Observations  0
Protection                                                   Compressed            NO
Data Set Type                                                Sorted                YES
Label                Metadata for all SAS data sets in SASHELP
Data Representation  WINDOWS_64
Encoding             wlatin1  Western (Windows)

                                  Engine/Host Dependent Information

Data Set Page Size          65536
Number of Data Set Pages    104
First Data Page             1
Max Obs per Page            151
Obs in First Data Page      145
Number of Data Set Repairs  0
ExtendObsCounter            YES
Filename                    S:\Projects\NH-COMPARE\Data_From_CMS\Monthly Processing\Files20150301\sashelp_d
Release Created             9.0401M1
Host Created                X64_ES08R2

                Alphabetic List of Variables and Attributes

 #    Variable    Type    Len    Format         Label

12    COMPRESS    Char      8                   Compression Routine
```

Output 1. Contents listing output from PROC DATASETS



LIBNAME	MEMNAME	NAME	TYPE	LENGTH	LABEL	FORMAT	NOBS	CRDATE
SASHELP	AACOMP	key	2	60			1544	05DEC13:02:13:4
SASHELP	AACOMP	lineno	1	4			1544	05DEC13:02:13:4
SASHELP	AACOMP	locale	2	5			1544	05DEC13:02:13:49
SASHELP	AACOMP	text	2	1200			1544	05DEC13:02:13:4
SASHELP	AARFM	key	2	60			61	05DEC13:02:13:5
SASHELP	AARFM	lineno	1	4			61	05DEC13:02:13:5
SASHELP	AARFM	locale	2	5			61	05DEC13:02:13:5
SASHELP	AARFM	text	2	1200			61	05DEC13:02:13:
SASHELP	ADSMSG	LEVEL	2	1			426	13:01:33:

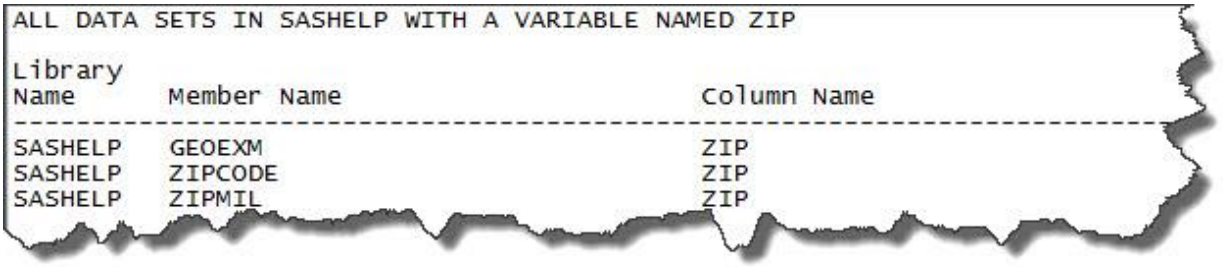
Output 2. Test print of output data set created by PROC DATASETS

RESOURCE #3. SAS DICTIONARY TABLES

There are many excellent papers on SAS DICTIONARY TABLES and PROC SQL and I urge SAS users to take advantage of the very useful information in them. Like PROC DATASETS, the utility of SAS DICTIONARY TABLES goes way beyond documentation and metadata purposes. In example 3, I demonstrate how to use dictionary tables to find all data sets in a library containing a variable named ZIP. Dictionary tables are accessed using PROC SQL.

```
* RESOURCE 3: SAS DICTIONARY TABLES;
/* select data sets in a library with the zip variable */
PROC SQL;
    TITLE1 'ALL DATA SETS IN SASHELP WITH A VARIABLE NAMED ZIP';
    SELECT LIBNAME, MEMNAME, NAME FROM DICTIONARY.COLUMNS
    WHERE NAME='ZIP' and libname='SASHELP';
QUIT;
```

The listing from PROC SQL is shown in ***Output 3***. We can see that three data sets in the SASHELP library have a variable named ZIP. We could have selected additional variables such as type, index information, and sort information to ensure any merges would be successful, and we could have created a permanent data set to do further manipulations, as SAS Dictionary Tables "disappear" when a SAS session ends.



ALL DATA SETS IN SASHELP WITH A VARIABLE NAMED ZIP

Library Name	Member Name	Column Name
SASHELP	GEOEXM	ZIP
SASHELP	ZIPCODE	ZIP
SASHELP	ZIPMIL	ZIP

Output 3. Listing of SAS Dictionary Information produced by PROC SQL

RESOURCE #4. SASHELP VIEWS

SASHELP views are analogous to dictionary tables. They are SAS views based on the same metadata that the dictionary tables are based on. They may be accessed using PROC SQL or with data step(s).

```
* RESOURCE 4: SASHELP VIEWS;
PROC SQL;
    CREATE TABLE LIB.SASHELP_ZIPCODE_CDBK AS
        SELECT NAME, TYPE, LENGTH, FORMAT, VARNUM, IDXUSAGE, SORTEDBY, LABEL
        FROM SASHELP.VCOLUMN
        WHERE LIBNAME='SASHELP' AND MEMNAME='ZIPCODE';
QUIT;

PROC PRINT DATA=LIB.SASHELP_ZIPCODE_CDBK (OBS=10) NOOBS;
RUN;
```

The listing from PROC SQL is shown in ***Output 4***. We could have used a data step in this case, but PROC SQL is much more efficient. We have created a permanent data set as the basis of a data dictionary, as the SAS view will not be accessible once our SAS session has ended.

SGF 2015 - SAMPLES

name	type	length	format	varnum	idxusage	sortedby	label
CITY2	char	64		1	COMPOSITE	0	Clean CITY name for geocoding
STATENAME2	char	64		2	COMPOSITE	0	Clean STATENAME for geocoding
ZIP	num	8	Z5.	3	SIMPLE	1	The 5-digit ZIP Code
Y	num	8	11.6	4		0	Latitude (degrees) of the center (centroid) c
X	num	8	11.6	5		0	Longitude (degrees) of the center (centroid) c
ZIP_CLASS	char	1		6		2	ZIP Code classification:P=PO Box U=Unique zip
CITY	char	35		7		0	Name of city/org
STATE	num	8		8		0	Two-digit number (FIPS code) for state/territ
STATECODE	char	2		9	COMPOSITE	0	Two-letter abbrev. for state name.
STATENAME	char	25		10		0	Full name of state/territory

Output 4. Listing of SASHELP View Information produced by PROC SQL

RESOURCE #5. %SYSFUNC

%SYSFUNC (or %QSYSFUNC) is a macro function that allows you to access SAS language functions regardless of the system environment you are operating in. For example, %IF %SYSFUNC(EXIST(infi)) %THEN %DO; . . . %END; sets up conditional processing based on infi's existence. Selected SAS® functions that can be used with %SYSFUNC: Verify existence of a file; open a file; get information about a file, delete a file, close a file, read a host-specific option. Unlike PROC CONTENTS, PROC DATASETS, Dictionary Tables, and SASHELP views, %SYSFUNC does produce tables and views, but instead uses macro processing to obtain SAS metadata and turn selected items into macro variables. These macro variables can be used in a number of ways. They can be used to create variables that can be used for documentation, to support conditional processing based on whether a file exists or not or whether a variable is character or numeric, to add documentation details to data set labels, variable labels, and catalog descriptions, what user name last run a program, etc. It is a very powerful tool and can be used in %LET statements, in titles/footnotes/ODS text/descriptions as macro variables, and incorporated into macro routines. Consider the following title statement:

```
TITLE1 " Created on &SYSDATE - &SYSTIME - by &SYSUSERID - using
%SYSFUNC(GETOPTION(SYSIN))";
```

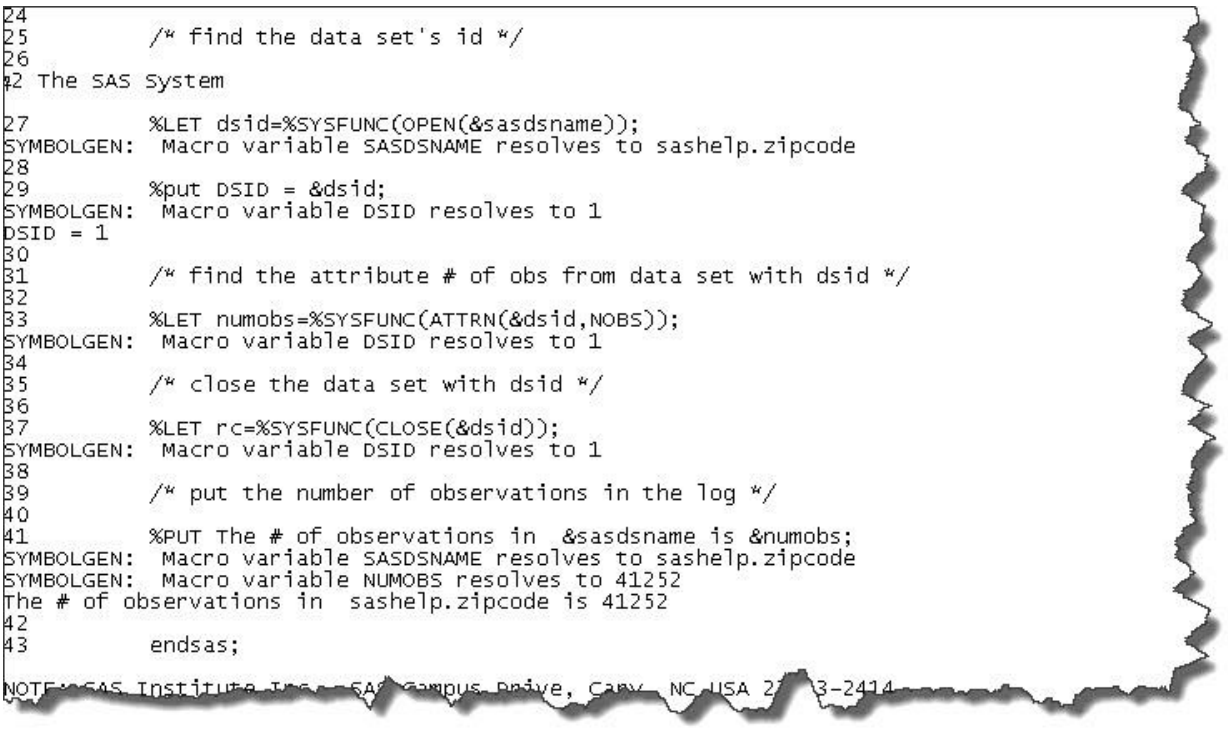
This single statement provides information to both the log and list as to when the program was last run, who ran it, and what the program name is. Similar text can be used to describe the data set in a data set label, describe variables, and so on.

The example shown is to write the number of observations in a data set to the log. Why bother? We perform monthly processing of data, and just once (last month) a data set was created with 0 observations, an error which did not trigger an ERROR statement in the log and had a domino effect on our processing routine. Early warning with a few %LET statements written to the log is a small price to pay!

```
* RESOURCE 5: %SYSFUNC;
/* find the number of observations in a data set */

/* check to see that the data set exists */

%LET dsexist=%SYSFUNC(EXIST(sashelp.zipcode));
%PUT &dsexist;

/* identify the data set */

%LET sasdsname=sashelp.zipcode;

/* find the data set's id */

%LET dsid=%SYSFUNC(OPEN(&sasdsname));

%put DSID = &dsid;

/* find the attribute # of obs from data set with dsid */
```

```
%LET numobs=%SYSFUNC(ATTRN(&dsid,NOBS));

/* close the data set with dsid */

%LET rc=%SYSFUNC(CLOSE(&dsid));

/* put the number of observations in the log */

%PUT The # of observations in  &sasdsname is &numobs;
```

A snippet from the log is shown in ***Output 5***.



```
24
25          /* find the data set's id */
26
2 The SAS System

27          %LET dsid=%SYSFUNC(OPEN(&sasdsname));
SYMBOLGEN:  Macro variable SASDSNAME resolves to sashelp.zipcode
28
29          %put DSID = &dsid;
SYMBOLGEN:  Macro variable DSID resolves to 1
DSID = 1
30
31          /* find the attribute # of obs from data set with dsid */
32
33          %LET numobs=%SYSFUNC(ATTRN(&dsid,NOBS));
SYMBOLGEN:  Macro variable DSID resolves to 1
34
35          /* close the data set with dsid */
36
37          %LET rc=%SYSFUNC(CLOSE(&dsid));
SYMBOLGEN:  Macro variable DSID resolves to 1
38
39          /* put the number of observations in the log */
40
41          %PUT The # of observations in  &sasdsname is &numobs;
SYMBOLGEN:  Macro variable SASDSNAME resolves to sashelp.zipcode
SYMBOLGEN:  Macro variable NUMOBS resolves to 41252
The # of observations in  sashelp.zipcode is 41252
42
43          endsas;

NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
```

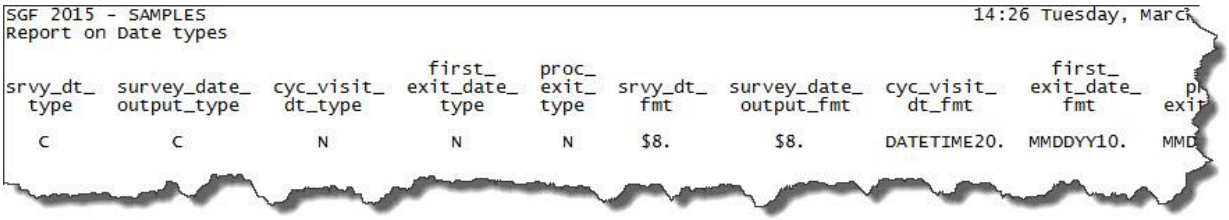
Output 5. Log with data set level information produced by %SYSFUNC

RESOURCE #6. "V" FUNCTIONS

"V" functions are analogous to %SYSFUNC macro functions, but are less powerful, specific to variables and used within the SAS data step. Like %SYSFUNC, it is a good resource for enabling conditional processing based on variable type, formats, etc.

```
* RESOURCE 6: "V" FUNCTIONS;
DATA dateformattest;
    LENGTH srvy_dt_type  . . . $ 1
           srvy_dt_fmt   . . .$ 32;
    SET defs.alltexts_indx20150301;
    IF _n_=1;

    srvy_dt_type=VTYPE(srvy_dt); .   . .
    srvy_dt_fmt=VFORMAT(srvy_dt); . . .
 RUN;
```

The listing from PROC PRINT of the variables created using a data step and "V" functions is shown in ***Output 6***. This could prevent one of those "gotchas" where dates come in as numeric one month and character another, but with the same variable names, or come in as DATETIME20. one month and MMDDYY10. another month.



SGF 2015 - SAMPLES 14:26 Tuesday, March
Report on Date types

srvy_dt_ type	survey_date_ output_type	cyc_visit_ dt_type	first_ exit_date_ type	proc_ exit_ type	srvy_dt_ fmt	survey_date_ output_fmt	cyc_visit_ dt_fmt	first_ exit_date_ fmt	pr exit
C	C	N	N	N	$8.	$8.	DATETIME20.	MMDDYY10.	MMD

Output 6. Print out of date variable types and formats produced using "V" functions

CONCLUSION

If you are looking to find SAS metadata, there's more than one way to skin a cat! Take advantage of the master procedure PROC DATASETS (as well as PROC CONTENTS), SAS Dictionary Tables and SASHELP.VIEWS, "V" functions and information gleaned from using the powerful %SYSFUNC to create user documentation, generate SAS code, perform conditional processing and GET INPUT!

REFERENCES

Carey, Helen and Carey, Ginger, 2011. "Tips and Techniques for the SAS Programmer!" Proceedings of SAS Global Forum 2011.

Crawford, Peter, 2013. "A Day in the Life of Data – Part 3." Proceedings of SAS Global Forum 2013.

Fraeman, Kathy Hardis, 2008. "Get into the Groove with %SYSFUNC: Generalizing SAS® Macros with Conditionally Executed Code." Proceedings of NESUG 2008.

Hadden, Louise, 2014. "Build your Metadata with PROC CONTENTS and ODS OUTPUT", Proceedings of SAS Global Forum 2014.

Huang, Chao, 2014. "Top 10 SQL Tricks in SAS®." Proceedings of SAS Global Forum 2014.

Karafa, Matthew T., 2012. "Macro Coding Tips and Tricks to Avoid "PEBCAK" Errors." Proceedings of SAS Global Forum 2012.

Kuligowski, Andrew T. and Shankar, Charu, 2013. "Know Thy Data: Techniques for Data Exploration." Proceedings of SAS Global Forum 2013.

Lafler, Kirk Paul, 2014. "Powerful and Hard-to-find PROC SQL Features." Proceedings of SAS Global Forum 2014.

Murphy, William C., 2013. "What's in a SAS® Variable? Get Answers with a V!" Proceedings of SAS Global Forum 2013.

Raithel, Michael A., 2011. "PROC DATASETS: the Swiss Army Knife of SAS® Procedures." Proceedings of SAS Global Forum 2011.

Thornton, Patrick, 2011. "SAS® DICTIONARY: Step by Step." Proceedings of SAS Global Forum 2011.

Zhang, Jingxian, 2012. "Techniques for Generating Dynamic Code from SAS® Dictionary Tables." Proceedings of SAS Global Forum 2012.

ACKNOWLEDGMENTS

The author gratefully acknowledges the helpful work of Kathy Fraeman, Michael Raithel, Patrick Thornton and Kirk Paul Lafler, among others.

CONTACT INFORMATION

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

Name: Louise Hadden
E-mail: Louise_Hadden@abtassoc.com