RIV-29

Build your Metadata with PROC CONTENTS and ODS OUTPUT

Louise S. Hadden, Abt Associates Inc.

ABSTRACT

Simply using an ODS destination to replay PROC CONTENTS output does not provide the user with attractive, usable metadata. Harness the power of SAS® and ODS output objects to create designer multi-tab metadata workbooks with the click of a mouse!

INFORMATION OVERLOAD

SAS procedures can convey an enormous amount of information – sometimes more information than is needed. By manipulating procedural output and ODS output objects, we can pick and choose just the information we want to see. We can then harness the power of SAS reporting procedures and ODS to present the information we want to display accurately and attractively. We wanted to create a metadata Excel® workbook with an index of all data sets delivered with separate variable level tabs for each data set to accurately describe a monthly data submission to our client. The challenge was how to accomplish this in a single SAS program without post-editing. This paper and presentation will demonstrate how we met our challenge.

DISCOVERING ODS OUTPUT OBJECTS

Most SAS procedures generate ODS objects behind the scenes. SAS uses these objects in conjunction with style templates that have custom "buckets" for certain types of output to produce the output we see in all destinations (including the SAS listing).

Use the ODS TRACE command to identify ODS output objects. Choose the procedure that you are using, and "surround" the procedure (including any ODS GRAPHICS calls, etc. you may be using) with ODS TRACE and ODS TRACE OFF commands.

```
ODS TRACE ON / LABEL;
ODS RTF FILE='yourfilename.rtf' PATH=ODSout STYLE=styles.journal2;
ODS GRAPHICS ON;
. . . your procedures here . . .
ODS RTF CLOSE;
ODS GRAPHICS OFF;
ODS TRACE OFF;
```

If you are running interactively, you will see the ODS objects generated in the results window on the left, and can select, view and save these objects. In some cases, you can manipulate the objects. ODS trace information will be located in your SAS log. Although the ODS objects can be viewed in the results window, you will not get enough information about the objects to truly customize your output, so the log is essential for both interactive and batch users.

If you are running any SG procedures, use the following command before running the procedures.

```
ODS LISTING SGE=ON;
```

This creates an editable (using the ODS graphics editor) file which allows you to change titles, background colors, etc. in your graphs which you can then save.

If you run in batch mode, output from the ODS TRACE command will be included in your log file.

UTILIZING ODS OUTPUT OBJECTS

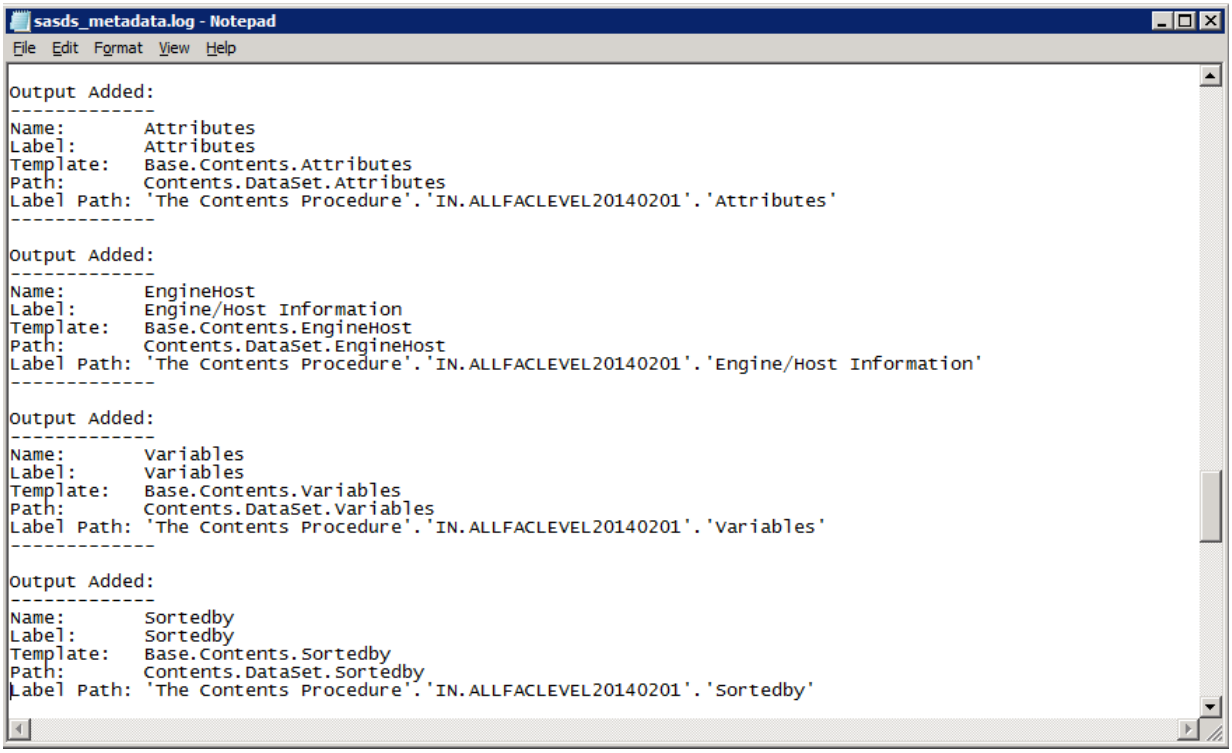
We wished to create an Excel workbook with an index tab with basic information about all data sets being delivered and documented for a given month in a given domain (files to be uploaded to DATA.MEDICARE.GOV), plus tabs for each data set with selected details on variables. In this case, we will be using the CONTENTS procedure in conjunction with ODS output objects. PROC CONTENTS does create an output data set (OUT=); however, it does not contain one desired piece of information, the variables that a data set are sorted by (if any). It also is rectangular,

meaning that many variables on a system level are included on every record. ODS RTF or ODS HTML calls sandwiched around PROC CONTENTS also yields a lot of unnecessary information and would require a lot of post-editing. A macro was designed that produces a PROC CONTENTS for each data set, outputs ODS output objects, manipulates the objects and outputs two temporary data sets (a header line for the index tab, and a variable listing) for each data set.

```
ODS TRACE ON / LABEL;

ODS RTF FILE='contents_allfaclevel.rtf' PATH=ODSout STYLE=styles.journal2;

ODS GRAPHICS ON;

    PROC CONTENTS DATA=dd.allfaclevel20140201;

    RUN;

ODS RTF CLOSE;

ODS GRAPHICS OFF;

ODS TRACE OFF;
```

EXPLORING ODS OUTPUT OBJECTS CREATED BY PROC CONTENTS

After running the code snippet above we can review our log and identify all the output objects generated by the contents procedure and their system names. Note that the output objects may vary! For example, SAS/STAT®'s PROC SURVEYFREQ produces different output objects based on whether you are running a one-way table or crosstabs, and may NOT generate chi-squared statistics if there are any 0 cells in a crosstab. In our case, if a data set is not sorted, PROC CONTENTS will not generate ODS output objects relating to sorting (SORTEDBY). If you want to report on sorting, you may need to have conditional processing that will indicated that a dataset is not sorted if there are no ODS output objects related to sorting.



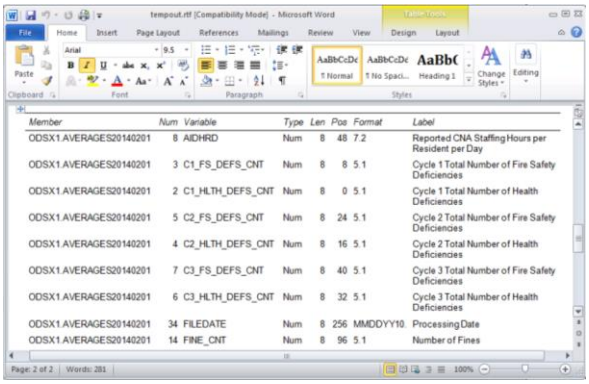
We can see that PROC CONTENTS generated many temporary ODS output objects (datasets) that go into the "print" output: ATTRIBUTES, VARIABLES, SORTEDBY and ENGINEHOST. These datasets are not necessarily on the same level and in order to create a single data set from the various pieces, more exploration is needed. To save these temporary files to either work or permanent data sets, use the ODS OUTPUT statement.

```
ODS OUTPUT ATTRIBUTES=atr VARIABLES=var SORTEDBY=sortedby;

PROC CONTENTS DATA=&inlib..&infi ;

RUN;

ODS OUTPUT CLOSE;
```

```
PROC CONTENTS DATA=var VARNUM;
RUN;
PROC PRINT DATA=var (obs=10) NOOBS;
RUN; . . .
```

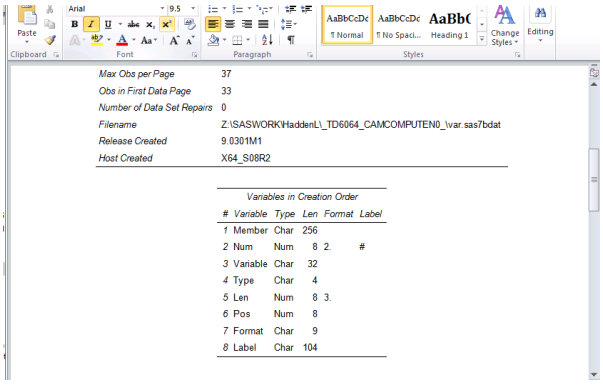
It is recommended that you do a test print of 10 observations even if you are working with a small data set, otherwise your output could be very large. 10 observations is generally enough to get a good picture. The contents is also very important as fields which might appear to be numeric may be character – SAS creates "print" variables for its output – and formatting give you valuable information for further manipulations. Variables within certain ODS output objects may not exist depending on the input data set(s). For example, in the case of PROC CONTENTS, you will not get the variable "informat" in the VARIABLES output object if no variables in the data set have informats.

Test print of VARIABLE ODS Output Object:



Member	Num	Variable	Type	Len	Pos	Format	Label
ODSX1.AVERAGES20140201	8	AIDHRD	Num	8	48	7.2	Reported CNA Staffing Hours per Resident per Day
ODSX1.AVERAGES20140201	3	C1_FS_DEFS_CNT	Num	8	8	5.1	Cycle 1 Total Number of Fire Safety Deficiences
ODSX1.AVERAGES20140201	2	C1_HLTH_DEFS_CNT	Num	8	0	5.1	Cycle 1 Total Number of Health Deficiences
ODSX1.AVERAGES20140201	5	C2_FS_DEFS_CNT	Num	8	24	5.1	Cycle 2 Total Number of Fire Safety Deficiences
ODSX1.AVERAGES20140201	4	C2_HLTH_DEFS_CNT	Num	8	16	5.1	Cycle 2 Total Number of Health Deficiences
ODSX1.AVERAGES20140201	7	C3_FS_DEFS_CNT	Num	8	40	5.1	Cycle 3 Total Number of Fire Safety Deficiences
ODSX1.AVERAGES20140201	6	C3_HLTH_DEFS_CNT	Num	8	32	5.1	Cycle 3 Total Number of Health Deficiences
ODSX1.AVERAGES20140201	34	FILEDATE	Num	8	256	MMDDYY10.	Processing Date
ODSX1.AVERAGES20140201	14	FINE_CNT	Num	8	96	5.1	Number of Fines

PROC CONTENTS of VARIABLE ODS Output Object:

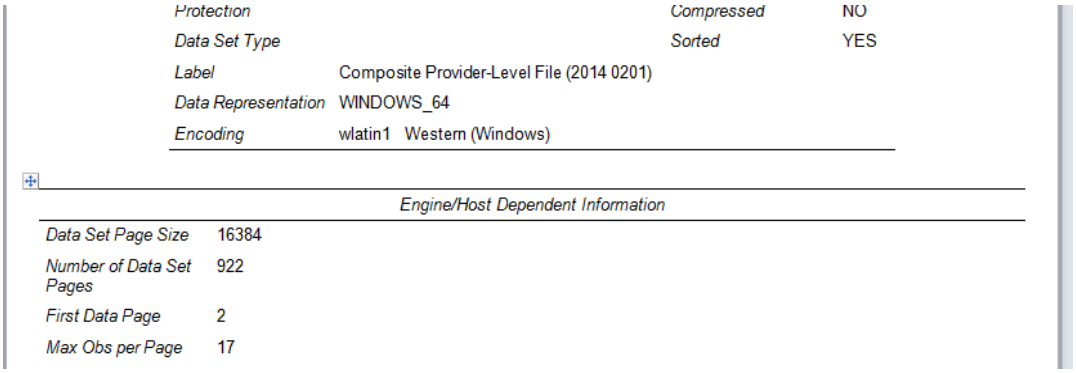


Max Obs per Page	37
Obs in First Data Page	33
Number of Data Set Repairs	0
Filename	Z:\SASWORK\HaddenL_TD6064_CAMCOMPUTEN0_\var.sas7bdat
Release Created	9.0301M1
Host Created	X64_S08R2

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	Member	Char	256		
2	Num	Num	8	2.	#
3	Variable	Char	32		
4	Type	Char	4		
5	Len	Num	8	3.	
6	Pos	Num	8		
7	Format	Char	9		
8	Label	Char	104		

Snippet of RTF output produced by an ODS sandwich around PROC CONTENTS statement: Note separate sections, not easy to parse out desired information.



Protection		Compressed	NO
Data Set Type		Sorted	YES
Label	Composite Provider-Level File (2014 0201)		
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	16384
Number of Data Set Pages	922
First Data Page	2
Max Obs per Page	17

MANIPULATING ODS OUTPUT OBJECTS

Reviewing all the output above allows us to pick and choose items and records from various output data sets, and merge them for reporting. The reasoning behind the test prints and PROC CONTENTS outputs becomes clear as it is vital to understand how the output objects mesh with one another. To enhance your final output, you can merge a descriptive data set in with longer variable descriptions and formatting variables that you might not want to carry in an analytic data set (for example, variables to create shading, italicizing, or bolding.) You can also create macro variables for printing from some ODS output objects using data steps and CALL SYMPUT. In this case, I used ATTRIBUTES, VARIABLES, and SORTEDBY output objects to construct two designer data sets for each data set to be documented. Note that because one of the input data bases is not sorted, we used conditional macro processing to create SORTEDBY if it does not exist.

```
/* first, set up a macro to (1) collect header level information for each
   DS and (2) collect variable level information for each ds */

%MACRO cont2(runnum,inlib,infi,headfi);

ODS OUTPUT ATTRIBUTES=atr
           VARIABLES=var
           SORTEDBY=sortedby;

PROC CONTENTS DATA=&inlib..&infi ;
RUN;

PROC SORT DATA=var;
    BY num;
RUN;

DATA &infi;
    SET var (drop=member pos);
    RENAME num=varnum
           variable=name
           len=length;
RUN;

%IF %SYSFUNC(EXIST(sortedby)) %then %do;

/* if it exists, go ahead and make our day */

DATA sortvar (KEEP=sortedby);
    LENGTH sortedby $ 1000;
    SET sortedby (WHERE=(label1='Sortedby'));
    sortedby=cvalue1;
RUN;

%END;

%ELSE %DO;  /* so it doesn't exist, we make it */

DATA sortvar;
    LENGTH sortedby $ 1000;
    sortedby='Ordered by Nation, then State';

RUN;

%END;

DATA sorted (KEEP=sorted);
    SET atr (WHERE=(label1='Data Set Type'));
    sorted=cvalue2;
RUN;

DATA memlabel (KEEP=memlabel);
    SET atr (WHERE=(label1='Label'));
    memlabel=cvalue1;
RUN;

data nvars (KEEP=nvars);
    SET atr (WHERE=(label1='Member Type'));
    nvars=INPUT(cvalue2,11.);
    FORMAT nvars comma11.;
RUN;

data member (KEEP=memname nobs);
    SET atr (WHERE=(label1='Data Set Name'));
    memname=cvalue1;
    nobs=INPUT(cvalue2,11.);
    FORMAT nobs comma11.;
RUN;
```

```
/* note lack of merge by variable deliberate */
DATA &headfi;
    MERGE member memlabel nvars sorted sortvar;
RUN;

%MEND cont2;

%cont2(1,in,allfaclevel&fileyear.&filedate.,header1);
%cont2(3,ODSx1,qualitymsr&fileyear.&filedate.,header3);
%cont2(2,ODSx1,alldeficiencies&fileyear.&filedate.,header2);
%cont2(4,ODSx1,ownership&fileyear.&filedate.,header4);
%cont2(5,ODSx1,penalties&fileyear.&filedate.,header5);
%cont2(6,ODSx1,averages&fileyear.&filedate.,header6);
%cont2(7,mds,nhqi_website_&QMlastQ._&filedate.,header7);
%cont2(8,mds,nhqi_website_q3_&filedate.,header8);

/* now build the header file */

DATA index;
    LENGTH memlabel sortedby $ 120 sorted $ 3;
    SET header1-header8;
RUN;

/* continued */
```

REPORTING ON DESIGNER DATA SETS BUILT FROM ODS OUTPUT OBJECTS

Our header file and individual data set variable listings have been built, and now it is time to create a multi-tab Excel workbook. The best way to do this and have some control over formatting is to use the ODS TAGSETS.EXCELXP tagset. We can name our tabs and set column widths within SAS for individual worksheets using tagset options: sheet_name names each tab, and Absolute_Column_Width sets the width for each column. We set and unset the sheet interval as table to generate a tab for each PROC PRINT.

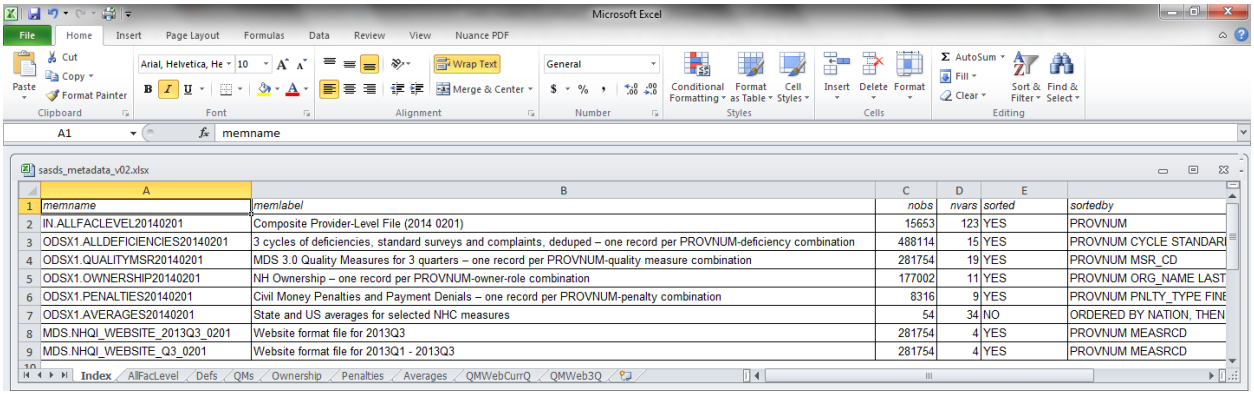
```
/* open an Excel workbook created via ODS TAGSETS.EXCELXP */

ODS TAGSETS.EXCELXP FILE=".\sasds_metadata_v&SASver..xml"
OPTIONS(orientation="Landscape") STYLE=styles.journal ;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="32,90,10,10,10,60");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Index" );
PROC PRINT DATA=index NOOBS;
    VAR memname memlabel nobs nvars sorted sortedby;
FORMAT nobs nvars comma11.;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="AllFacLevel" );
PROC PRINT DATA=allfaclevel&fileyear.&filedate. NOOBS;
    VAR varnum name label type length format;
RUN;

. . . [repetitive code omitted – ask author for full code sample if needed] . . .

ODS TAGSETS.EXCELXP CLOSE;
QUIT;
```

XML TO XLS(X)

If you looked carefully at the code snippet above, you noticed that the TAGSETS.EXCELXP generates XML and not XLS or XLSX. We use DDE from within our SAS program to "Save As", as we work in a server environment and other options for saving as do not work for us. Once this has successfully completed, then we delete the XML version of the file.

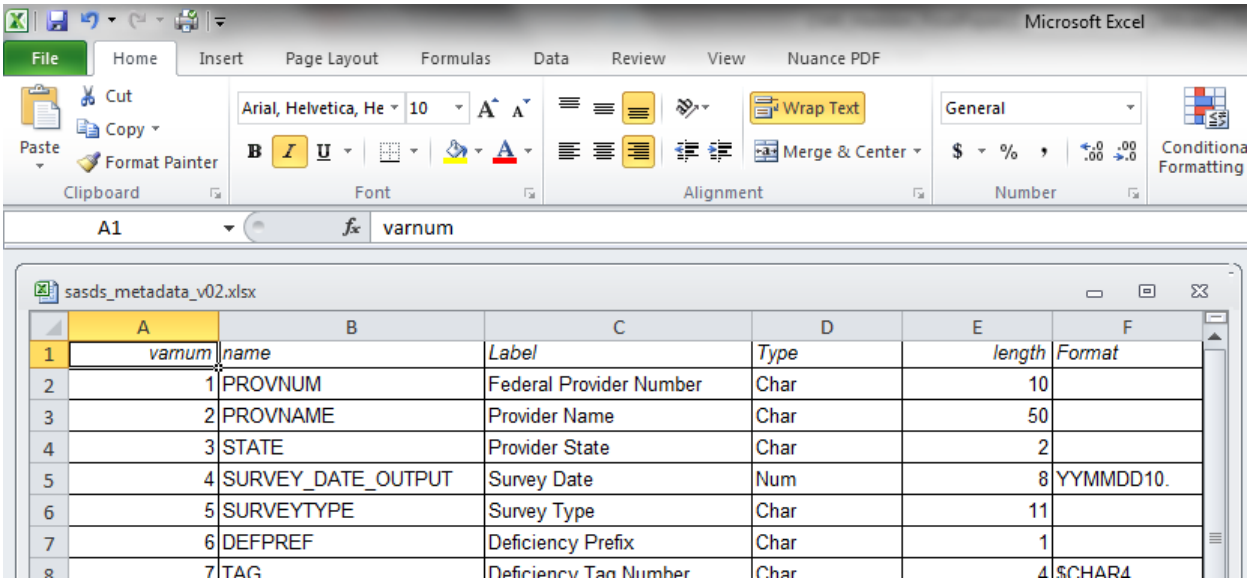
```
/* make sure Excel is open before running program */

OPTIONS NOXWAIT NOXSYNC;

* This next routine starts the Excel application.   *;
FILENAME CMDS DDE 'excel|system';
X "'C:\Program Files\Microsoft Office\Office14\excel.exe'";

DATA _null_;
    x=sleep(3);
RUN;

FILENAME cmds dde 'excel|system';
DATA _null_;
FILE cmds;
X=sleep(10);
PUT "[open(""S:\Projects\NH-COMPARE\Data_From_CMS\Monthly
    Processing\Files&fileyear.&filedate.\5Output\DataMedicareGov\
    sasds_metadata_v&SASver..xml"")]";
PUT '[ERROR("FALSE")]';
PUT "[SAVE.AS(""S:\Projects\NH-COMPARE\Data_From_CMS\Monthly
    Processing\Files&fileyear.&filedate.\5Output\DataMedicareGov\
    sasds_metadata_v&SASver..xlsx"" ,51)]";
X=sleep(2);
PUT '[close("false")]';
PUT '[quit]';
RUN;

X "del sasds_metadata_v&SASver..xml";
X "exit";
```

Screenshot of Index Tab in final workbook:



	memname	memlabel	nobs	nvars	sorted	sortedby
2	IN.ALLFACLEVEL20140201	Composite Provider-Level File (2014 0201)	15653	123	YES	PROVNUM
3	ODSX1.ALLDEFICIENCIES20140201	3 cycles of deficiencies, standard surveys and complaints, deduped – one record per PROVNUM-deficiency combination	488114	15	YES	PROVNUM CYCLE STANDARI
4	ODSX1.QUALITYMSR20140201	MDS 3.0 Quality Measures for 3 quarters – one record per PROVNUM-quality measure combination	281754	19	YES	PROVNUM MSR_CD
5	ODSX1.OWNERSHIP20140201	NH Ownership – one record per PROVNUM-owner-role combination	177002	11	YES	PROVNUM ORG_NAME LAST
6	ODSX1.PENALTIES20140201	Civil Money Penalties and Payment Denials – one record per PROVNUM-penalty combination	8316	9	YES	PROVNUM PNLTY_TYPE FINE
7	ODSX1.AVERAGES20140201	State and US averages for selected NHC measures	54	34	NO	ORDERED BY NATION, THEN
8	MDS.NHQI_WEBSITE_2013Q3_0201	Website format file for 2013Q3	281754	4	YES	PROVNUM MEASRCD
9	MDS.NHQI_WEBSITE_Q3_0201	Website format file for 2013Q1 - 2013Q3	281754	4	YES	PROVNUM MEASRCD

Screenshot of Defs Tab in final workbook:



	A	B	C	D	E	F
1	*varnum*	*name*	*Label*	*Type*	*length*	*Format*
2	1	PROVNUM	Federal Provider Number	Char	10	
3	2	PROVNAME	Provider Name	Char	50	
4	3	STATE	Provider State	Char	2	
5	4	SURVEY_DATE_OUTPUT	Survey Date	Num	8	YYMMDD10.
6	5	SURVEYTYPE	Survey Type	Char	11	
7	6	DEFPREF	Deficiency Prefix	Char	1	
8	7	TAG	Deficiency Tag Number	Char	4	$CHAR4

CONCLUSION

Using the four basic concepts outlined below you can quickly and easily create your metadata with the click of a mouse.

1. Identify / locate your ODS output object(s) using ODS TRACE or output data set(s) .
2. Analyze your ODS output object(s) or output data set(s) using basic SAS procedures and review.
3. Manipulate your ODS output object(s) or output data set(s) using SAS data steps and/or procedures.
4. Report on your final data set.

Using ODS output objects and output data sets can be an enormous time saver. Application of this technique is not limited to metadata: the screenshot below was created by manipulating a number of different ODS output objects including Chi Squared statistics as well as frequency output from the PROC SURVEYFREQ procedure run on a large complex data set. No post-processing was required.

Table 1. Selected Characteristics by BMI Category

Characteristic	*Total 2005*	*Total 2005*	*Chi Squared P-Value*	*Underweight (BMI<18.5)*	*Normal Weight (BMI 18.5-24.9)*	*Overweight (BMI 25-29.9)*	*Obese (BMI 30+)*
Total Sample	**15,195 (946,008)**	**100.0±0.00**	**NA**	**1.2±0.14**	**38.3±0.82**	**47.6±0.68**	**12.9±0.55**
Sex							
Men	11,395 (804,888)	85.1±0.71	<.0001	0.9±0.16	35.2±0.94	50.2±0.79	13.7±0.61
Women	3,800 (141,120)	14.9±0.71	<.0001	2.8±0.30	56.1±1.24	32.9±1.01	8.2±0.63
Age							
17-20	1,187 (130,680)	13.8±0.99	<.0001	1.6±0.40	53.8±1.95	37.9±1.78	6.7±1.35
21-30	6,180 (481,590)	50.9±1.21	<.0001	1.5±0.21	42.4±0.98	44.7±0.96	11.4±0.58
31-39	4,610 (230,112)	24.3±1.05	<.0001	0.6±0.17	26.9±1.22	54.8±0.91	17.7±0.85
40+	3,218 (103,627)	11.0±0.75	<.0001	0.7±0.20	25.2±1.15	57.2±1.22	16.9±0.98
Educational Attainment							
High School or Less	4,409 (352,500)	37.3±1.56	<.0001	1.1±0.26	43.9±1.26	41.8±0.95	13.3±0.93
Some College	6,146 (380,144)	40.2±1.26	<.0001	1.6±0.20	34.5±0.80	49.7±0.79	14.2±0.67

With ODS output objects / output data sets and SAS, the sky is the limit!

REFERENCES AND RECOMMENDED READING

Guan, Calvin and Hadden, Louise. "EXCELing in DDE: Unlock Useful Tools for Processing Excel®". *Proceedings of NorthEast SAS Users Group 2013 Conference*. September, 2013.

Hadden, Louise. "With a Trace: Making Procedural Output and ODS Output Objects Work For You". *Proceedings of SAS Global Forum 2013 Conference.* May 2013.

Zender, Cynthia. "The Greatest Hits: ODS Essentials Every User Should Know". *Proceedings of SAS Global Forum 2011 Conference.* April 2011.

ACKNOWLEDGEMENTS

The author would like to express her appreciation for the inspiration, creativity and help provided by Cynthia Zender of SAS and colleagues Christianna Williams, Calvin Guan, and Alan White.

CONTACT INFORMATION

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

Louise Hadden louise_hadden@abtassoc.com

Code samples (including for the table snippet shown above) are available upon request.

APPENDIX A: Full code to create a CONTENTS workbook of selected SAS data sets in the SASHELP folder.

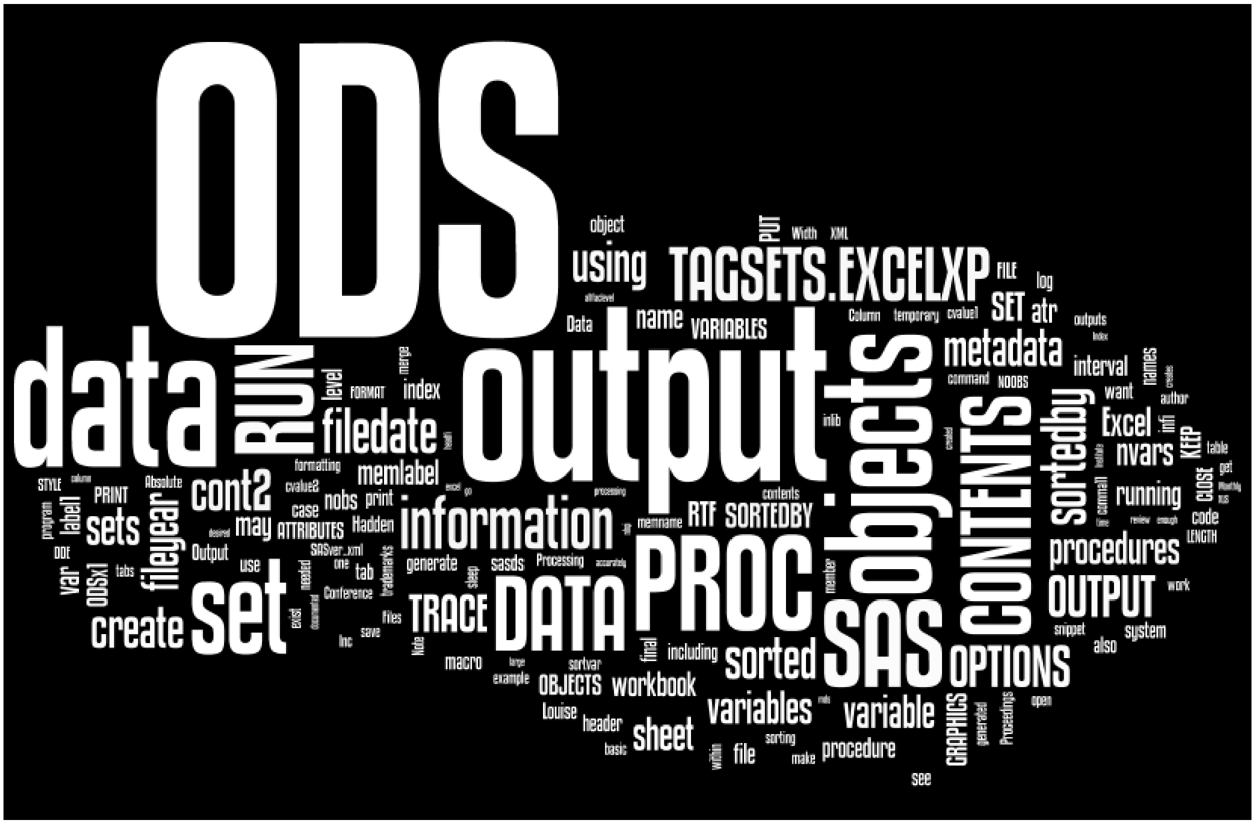
```
OPTIONS errorabend ;
OPTIONS OBS=MAX mprint  symbolgen nonumber noquotelenmax  ;
OPTIONS formchar="|----|+|---+=|-/\<>*";
RUN;

LIBNAME FMT ".";
LIBNAME library ".";
filename ODSout '.';
RUN;

 /*****************************************
 Program:   SASDS_METADATA_PUF.sas
 Purpose:
(1) Produce a metadata spreadsheet containing contents of SAS ds
    with a separate tab for each Data Set

 Ancestors:
 Notes:

 Input Data:  (1) (in SASHELP) class.sas7bdat
              (2) (in SASHELP) demographics.sas7bdat
              (3) (in SASHELP) cars.sas7bdat
              (4) (in SASHELP) electric.sas7bdat
              (5) (in SASHELP) fish.sas7bdat
              (6) (in SASHELP) gas.sas7bdat
              (7) (in SASHELP) heart.sas7bdat
              (8) (in SASHELP) lake.sas7bdat
              (9) (in SASHELP) mileages.sas7bdat
              (10) (in SASHELP) prdsale.sas7bdat
              (11) (in SASHELP) shoes.sas7bdat
              (12) (in SASHELP) snacks.sas7bdat
              (13) (in SASHELP) stocks.sas7bdat
              (14) (in SASHELP) zipcode.sas7bdat

 Output Data: (1) sasds_metadata_SASHELP.xlsx

 Created:  9/16/2013
 Author:  Louise S. Hadden
 Edited:

 *****************************************/

RUN;

ODS NOPROCTITLE;

/* first, set up a macro to (1) collect header level information for each DS and (2)
collect variable level information for each ds */

%MACRO cont2(runnum,inlib,infi,headfi);

ODS OUTPUT ATTRIBUTES=atr
           VARIABLES=var
           SORTEDBY=sortedby&infi.;

PROC CONTENTS DATA=&inlib..&infi ;
RUN;

PROC SORT DATA=var;
    BY num;
RUN;

DATA &infi;
    LENGTH format $ 32 label $ 200;  /* necessary to SET these variables up because
they don't exist */
```

```
    SET var (DROP=member pos);
    RENAME num=varnum
           variable=name
           len=length;
RUN;

  %IF %SYSFUNC(EXIST(sortedby&infi)) %THEN %DO;

  /* if it exists, go ahead and make our day */

DATA sortvar (KEEP=sortedby);
    LENGTH sortedBY $ 1000;
    SET sortedby&infi (WHERE=(label1='Sortedby'));
    sortedby=cvalue1;
RUN;

  %END;

  %ELSE %DO;  /* so it doesn't exist, we make it */

DATA sortvar;
    LENGTH sortedBY $ 1000;
    sortedby='NOT SORTED';

  RUN;

  %END;

DATA sorted (KEEP=sorted);
    SET atr (WHERE=(label1='DATA SET Type'));
    sorted=cvalue2;
RUN;

DATA memlabel (KEEP=memlabel);
    SET atr (WHERE=(label1='Label'));
    memlabel=cvalue1;
RUN;

DATA nvars (KEEP=nvars);
    SET atr (WHERE=(label1='Member Type'));
    nvars=input(cvalue2,11.);
    format nvars comma11.;
RUN;

DATA member (KEEP=memname nobs);
    SET atr (WHERE=(label1='DATA SET Name'));
    memname=cvalue1;
    nobs=input(cvalue2,11.);
    format nobs comma11.;
RUN;

/* note lack of merge BY variable deliberate */
DATA &headfi;
    MERGE member memlabel nvars sorted sortvar;
RUN;


%MEND cont2;

%cont2(1,sashelp,class,header1);
%cont2(2,sashelp,demographics,header2);
%cont2(3,sashelp,cars,header3);
%cont2(4,sashelp,electric,header4);
%cont2(5,sashelp,fish,header5);
%cont2(6,sashelp,gas,header6);
%cont2(7,sashelp,heart,header7);
%cont2(8,sashelp,lake,header8);
```

```
%cont2(9,sashelp,mileages,header9);
%cont2(10,sashelp,prdsale,header10);
%cont2(11,sashelp,shoes,header11);
%cont2(12,sashelp,snacks,header12);
%cont2(13,sashelp,stocks,header13);
%cont2(14,sashelp,zipcode,header14);

/* now build the header file */

DATA index;
    LENGTH memlabel sortedBY $ 120 sorted $ 3;
    SET header1-header14;
RUN;

PROC PRINT DATA=index NOOBS;
TITLE1 'Index File';

RUN;

ODS TAGSETS.EXCELXP file=".\sasds_metadata_SASHELP.xml"
OPTIONS(orientation="Landscape") style=styles.journal ;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="32,90,10,10,10,60");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Index" );
PROC PRINT DATA=index NOOBS;
    VAR memname memlabel nobs nvars sorted sortedby;
FORMAT nobs nvars comma11.;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Class" );
PROC PRINT DATA=class NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Demographics" );
PROC PRINT DATA=Demographics NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Cars" );
PROC PRINT DATA=cars NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Electric" );
PROC PRINT DATA=electric NOOBS;
    VAR varnum name label type length format;
RUN;
ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Fish" );
PROC PRINT DATA=fish NOOBS;
```

```
    VAR varnum name label type length format;
RUN;
ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Gas" );
PROC PRINT DATA=gas NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Heart" );
PROC PRINT DATA=heart NOOBS;
    VAR varnum name label type length format;
RUN;
ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Lake" );
PROC PRINT DATA=Lake NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Mileages" );
PROC PRINT DATA=mileages NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="PRDSale" );
PROC PRINT DATA=prdsale NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Shoes" );
PROC PRINT DATA=shoes NOOBS;
    VAR varnum name label type length format;
RUN;
ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Snacks" );
PROC PRINT DATA=snacks NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Stocks" );
PROC PRINT DATA=stocks NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP OPTIONS(Absolute_Column_Width="10,32,80,10,10,10");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="Table");
ODS TAGSETS.EXCELXP OPTIONS(sheet_interval="None");
```

```
ODS TAGSETS.EXCELXP OPTIONS(sheet_name="Zipcode" );
PROC PRINT DATA=zipcode NOOBS;
    VAR varnum name label type length format;
RUN;

ODS TAGSETS.EXCELXP CLOSE;
QUIT;

/* make sure Excel is open before running program */

OPTIONS noxwait noxsync;

 * This next routine starts the Excel application.   *;
FILENAME cmds dde 'excel|system';
X "'C:\Program Files\Microsoft Office\Office14\excel.exe'";

DATA _null_;
   x=sleep(3);
RUN;

FILENAME cmds dde 'excel|system';
DATA _null_;
FILE cmds;
X=sleep(10);
PUT "[open(""INSERTYOURPATHNAMEHERE\sasds_metadata_SASHELP.xml"")]";
PUT '[ERROR("FALSE")]';
PUT "[SAVE.AS(""INSERTYOURPATHNAMEHERE\sasds_metadata_SASHELP.xlsx"" ,51)]";
x=sleep(2);
PUT '[close("false")]';
PUT '[quit]';
RUN;

X "del sasds_metadata_SASHELP.xml";
X "exit";

ENDSAS;
```


ODS
output
data
RUN
objects
TAGSETS.EXCELXP
PROC
SAS
DATA
set
CONTENTS
information
metadata
sortedby
filedate
cont2
fileyear
sets
create
TRACE
OUTPUT
OPTIONS
procedures
sorted
variables
variable
sheet
workbook
using
name
VARIABLES
SET
atr
interval
Excel
nvars
running
memlabel
index
level
RTF
SORTEDBY
ATTRIBUTES
may
nobs
print
PRINT
label1
var
ODSx1
procedure
file
GRAPHICS
OBJECTS
macro
generate
sasds
also
system
snippet
code
KEEP
CLOSE
want
names
author
log
FILE
PUT
Width
XML
object
Data
Column
temporary
cvalue1
command
NOOBS
inlib
outputs
table
get
LENGTH
work
open
Proceedings
generated
see
make
sorting
contents
memname
including
final
header
Louise
basic
within
example
sortvar
Processing
files
save
Inc
Conference
tab
use
Output
Absolute
STYLE
FORMAT
formatting
cvalue2
case
Hadden
merge