CREATING OUTPUT DATA SETS FROM ALMOST ANY PROCEDURE USING THE OUTPUT DELIVERY SYSTEM


ABSTRACT
Few of SAS' many procedures provide the option to create an output data set. You have probably found this to be a shortcoming on many occasions. For years you have likely seen many examples of the wonderful HTML, RTF, and PDF results that you can produce with the Output Delivery System (ODS). What you may not know is that the ODS has an option that allows you to produce an output data set from most SAS procedures. I'm serious. With just the simple ODS OUTPUT statement, SAS will create a SAS data set from the procedure output. Within this paper, the author will present SAS code that will enable you to quickly and easily create an output SAS data set from many of your favorite procedures.

INTRODUCTION
How could SAS possibly improve on the procedures that we have come to know and love? Some of our favorite procedures utilize the OUT= option, such as the SUMMARY, REPORT, CONTENTS, and FREQ procedures, to create an output SAS data set. But, most procedures have not provided the user the means to create an output SAS data set. But the ODS changes that, as now you can create an output SAS data set from almost any procedure using the ODS OUTPUT statement. However, there are a few procedures that do not allow ODS to create an output SAS data set, including the REPORT, PRINT, and SUMMARY procedures. Hey, the ODS OUTPUT statement even supports SAS data set options, such as DROP and KEEP. By the way, in all of the SAS code examples below I am using the sample SAS data set named 'Baseball' that I created from an option in SAS/ASSIST.

SYNTAX
Let’s take a look at the syntax we will need.

ODS OUTPUT <action>;
ODS OUTPUT output-object-specification=<SAS-data-set>;

What? Is that it? A whole SAS conference paper on that little bit of syntax! Well, getting what you want is a little more involved. Let’s look a bit further.

Actions Speak Louder Than Words
The SAS on-line documentation for the Output Delivery System defines the three possible actions on the ODS OUTPUT statement as follows:

CLEAR sets the list for the Output destination to EXCLUDE ALL.
CLOSE closes the Output destination. When an ODS destination is closed ODS will not send output to that destination.
SHOW writes to the SAS log the current selection or exclusion list for the Output destination. If the list is the default list (EXCLUDE ALL), SHOW also writes the current overall selection or exclusion list.

Specifying an Output Object
The SAS on-line documentation for the Output Delivery System defines the output-object-specification as follows:

output-object<action>(MATCH_ALL=<macro-var-name> PERSIST=PROC | RUN>)

The output-object specification identifies one or more output objects to turn into a SAS data set. To specify an output object, you need to know what output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. We will examine the ODS TRACE statement in the next section. The SAS on-line documentation for the Output Delivery System states that you can specify an output object as:

- a full Path
- a partial Path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path.
- a Label (surrounded by quotation marks). The label is the same label you will see in the Results window tree.
- a Label Path.
- a partial Label Path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label.
- a mixture of labels and paths.
- any of these specifications followed by a pound sign (#) and a number. For example, TestsForLocation#3 refers to the third output object named TestsForLocation.

While undocumented, I have found that specifying the Name value also works.

SAS-data-set
The SAS-data-set specification identifies the output data set. You can use a one-level or two-level (using a LIBREF) name. You can also add SAS Data Set options, such as DROP and KEEP.

An example of what we have seen so far might look like the following:

```sas
ods output clear;
odsexample OneWayFreqs=out.freq(drop=table);
odsexample show;
```

IDENTIFYING AN OUTPUT OBJECT
We can use the ODS TRACE statement to have SAS tell us the values of the output objects. By default, the ODS TRACE statement does not identify the Label Path. However, adding the LABEL option causes the trace record
to identify the Label Path. Below is a SAS code example of a FREQ procedure using the ODS TRACE statement and the resulting SAS log.

```sas
ods trace on /label;
proc freq data=in.baseball;
tables no_home;
run;
```

Output Added:
-------------
Name: OneWayFreqs
Label: One-Way Frequencies
Template: Base.Freq.OneWayFreqs
Path: Freq.NO_HOME.OneWayFreqs
Label Path: 'The Freq Procedure'.'Table NO_HOME'.'One-Way Frequencies'
-------------

The ODS TRACE ON statement returns the values of the Name, Label, Path, and Label Path that we can use to specify the output object. Now that we know the values of the output object, we can specify that we want to use the ODS OUTPUT statement to create an output SAS data set. Using the Name value for the output object (look carefully at the ODS OUTPUT statement below and the results of the trace above) and using out.freq as the output SAS data set name, we have the code below.

```sas
ods output OneWayFreqs=out.freq;
proc freq data=in.baseball;
tables no_home;
run;
```

Note the result in the SAS log shown below.

```
NOTE: The data set OUT.FREQ has 37 observations and 7 variables.
```

That is all there is to it! Let's take a look at the contents properties of the OUT.FREQ SAS data set we just created.

```
WHICH IS BETTER
Earlier I mentioned that the ODS OUTPUT statement does not work with all procedures. For example, it does not work with the REPORT and SUMMARY procedures (two of my personal favorites). You might be thinking, both of those procedures support the OUT= options to create an output SAS data set and that is why the ODS OUTPUT statement does not work with them. Not quite. The OUT= option is not the key because the FREQ and MEANS procedures support the OUT= options and both work with the ODS OUTPUT statement. So, for those procedures that support the OUT= option and also work with the ODS OUTPUT statement, the question then is which method is better to create an output SAS data set? Below is the same code for the FREQ procedure example, but we have added the OUT= option.

```sas
ods output OneWayFreqs=out.freq;
proc freq data=in.baseball;
tables no_home / out=out.frq_out;
```

In this example, we are still creating OUT.FREQ using the ODS OUTPUT statement, but we are also creating OUT.FREQ_OUT using the OUT= option. Notice the SAS Log below.

```
NOTE: The data set OUT.FREQ has 37 observations and 7 variables.
NOTE: There were 322 observations read from the data set IN.BASEBALL.
NOTE: The data set OUT.FREQ_OUT has 37 observations and 3 variables.
```

Notice that in this example, both SAS data set created have the same number of observations, but not the same number of variables. The SAS data set created using the ODS OUTPUT statement contains four more variables than does the SAS data set created by the OUT= option. Now, take a look at the CONTENTS and the first five rows of the SAS data set created with the OUT= option. The FREQ procedure using the OUT= option has created only three variables: the variable we selected, plus the count and the percent. Also note that in the CONTENTS output each
variable is labeled. In contrast, the SAS data set created using the ODS OUTPUT statement also includes variables for a cumulative count and a cumulative percent.

In this case, the SAS data set created by the ODS OUTPUT statement (OUT.MEANS) contains 8 variables, while the SAS data set created by the OUT= option (OUT.MEANS_OUT) contains only 7 variables. Below is the CONTENTS output and the first 5 rows of the SAS data set created by the ODS OUTPUT statement.

Let's look at another example, this time using the MEANS procedure. First, we will use the ODS TRACE ON statement to identify the output objects.

Let's look at another example, this time using the MEANS procedure. First, we will use the ODS TRACE ON statement to identify the output objects.

Output Added:
------------
Name: Summary
Label: Summary statistics
Template: base.summary
Path: Means.Summary
Label Path: 'The Means Procedure'. 'Summary statistics'
------------

This time, we will use the Label to specify the output object in our code. (If you haven't seen the WAYS statement and the WAYS and LEVELS options, take a look at my SUGI 28 paper on new ways to summarize data.)

```sas
ods output "Summary statistics"=out.means;
proc means data=in.baseball missing;
  class team position;
  ways 2;
  var no_home;
  output out=out.means_out sum=no_home /ways levels;
run;
```

Then, notice the resulting SAS log.

```
NOTE: The data set OUT.MEANS has 260 observations and 8 variables.
NOTE: There were 322 observations read from the data set IN.BASEBALL.
NOTE: The data set OUT.MEANS_OUT has 260 observations and 7 variables.
```

Notice that the SAS data set created using the OUT= option contains the automatic variables _WAY_, _TYPE_, _LEVEL_ and _FREQ_, whereas the SAS data set created by the ODS OUTPUT statement does not. Also, the statistics created by the ODS OUTPUT statement are not present in the SAS data set created using the OUT= option.
So, when a procedure includes the OUT= option and is supported by the ODS OUTPUT statement, which method is better to create an output SAS data set? That may depend on what you need.

**KEEPING TRACK OF THE SELECTION LIST**
The ODS OUTPUT statement keeps a list of the objects selected for output. This is known as the ‘output selection list’. In the syntax presented earlier, we saw the ODS OUTPUT <action> statement. One of the actions is SHOW. Let’s add the SHOW action and see what the log reveals. First the SAS code:

```sas
ods output OneWayFreqs=out.freq;
ods output show;
proc freq data=in.baseball;
  tables no_home;
run;
ods output show;
```

And, then the SAS log:

```sas
ods output OneWayFreqs=out.freq;
ods output show;
Current OUTPUT select list is:
  1. OneWayFreqs=OUT.FREQ
proc freq data=in.baseball;
  tables no_home /out=out.freq_out;
run;
NOTE: The data set OUT.FREQ has 37 observations and 7 variables.
NOTE: There were 322 observations read from the data set IN.BASEBALL.
ods output show; /*selection list has cleared*/
Current OUTPUT exclude list is set to default value (ALL).
Current OVERALL select list is: ALL
```

Notice that the SHOW action writes to the SAS log the current output selection list before the output is created. In this example:

1. OneWayFreqs=OUT.FREQ

But, after the procedure has completed, ODS has cleared the output selection list. Issuing the SHOW action again writes to the SAS log the current output select list as:

Current OUTPUT exclude list is set to default value (ALL).
Current OVERALL select list is: ALL

This demonstrates that after the procedure creates the output SAS data set, the output selection list is reset to the default value of EXCLUDE ALL. There is a way to prevent the output selection list from being reset, as we will explore later.

**THE OPTIONS**
Harken back to the syntax:

```sas
output-object<(MATCH_ALL=<macro-var-name>
PERSIST=PROC | RUN)>=
```

Hopefully, you were wondering what the MATCH_ALL and PERSIST= are all about. Usually, in life there are options. SAS is no exception. When running multiple iterations of a procedure using only one ODS OUTPUT statement, the MATCH_ALL and PERSIST options can be useful. The MATCH_ALL=<macro-var-name> option causes ODS to create a new SAS data set for each output object. If you specify a value for macro-var-name, ODS stores a list of all the SAS data sets that it creates in the macro variable that you specify.

The PERSIST=PROC | RUN option determines when ODS closes any SAS data sets that it has created and when it removes output objects from the selection list for the Output destination. SAS offers two ways to use the PERSIST option: PROC and RUN. When using PROC, ODS retains the selection list of output definitions even if the procedure or DATA step ends, until you explicitly modify the selection list. When using RUN, ODS retains the selection list of output definitions and keeps the data sets that it is creating open even if the procedure or DATA step ends, until you explicitly modify the selection list.

If you are creating a single SAS data set, the ODS OUTPUT statement uses the name that you specify. If you are running multiple iterations of a procedure with only one ODS OUTPUT statement, ODS appends a digit to the name of the SAS data set. For example, if you specify “MEANS” as SAS-data-set name and you run three iterations of the MEANS procedure, ODS names the first SAS data set “MEANS”, the second “MEANS1”, and the third “MEANS2”.

Below is an example of running the MEANS procedure twice with one ODS OUTPUT statement. In the code below, we have used the Path value to specify the output object (refer to the ODS TRACE shown earlier). Notice the ODS OUTPUT SHOW statement before and after each procedure.

```sas
ods output Means.Summary(MATCH_ALL=
  PERSIST=PROC)=out.means;
ods output show;
proc means data=in.baseball missing;
  class team position;
  ways 2;
  var no_home;
run;
ods output show;
proc means data=in.baseball missing;
  class league team;
  ways 2;
  var no_home;
run;
```

Notice in the SAS log that the “Current OUTPUT select list” does not change before the first iteration of the MEANS
procedure, before the second iteration, or after the second iteration. The PERSIST=PROC option has kept SAS from clearing the selection list after each iteration. Also, notice the name of the second SAS data set created is OUT.MEANS1. The MATCH_ALL option has incremented the name of the SAS data set specified by adding a digit to the end of the SAS data set name.

```ods output Means.Summary(MATCH_ALL PERSIST=PROC)=out.means;
ods output show;
Current OUTPUT select list is:
1. Means.Summary(MATCH_ALL PERSIST=PROC)=OUT.MEANS
proc means data=in.baseball missing;
   class team position;
   ways 2;
   var no_home;
run;
NOTE: The data set OUT.MEANS has 260 observations and 10 variables.
NOTE: There were 322 observations read from the data set IN.BASEBALL.
ods output show;
Current OUTPUT select list is:
1. Means.Summary(MATCH_ALL PERSIST=PROC)=OUT.MEANS
proc means data=in.baseball missing;
   class league team;
   ways 2;
   var no_home;
run;
NOTE: The data set OUT.MEANS1 has 26 observations and 10 variables.
NOTE: There were 322 observations read from the data set IN.BASEBALL.
ods output show;
Current OUTPUT select list is:
1. Means.Summary(MATCH_ALL PERSIST=PROC)=OUT.MEANS
```

Remember the ODS OUTPUT <action> statement and the CLEAR and CLOSE options? These become useful when using the PERSIST option. As we see above, the “Current OUTPUT select list” does not clear after the procedure run. Issuing “ODS OUTPUT CLEAR” will clear the list and issuing “ODS OUTPUT CLOSE” will close the Output destination.

**PROCEDURES WITH MULTIPLE OUTPUT**

Some procedures, such as CORR and CONTENTS, produce multiple output objects. Using the ODS TRACE ON statement with the CONTENTS procedure yields the following:

```
Output Added:  
--------------
Name: Attributes
Label: Attributes
Template: Base.Contents.Attributes
Path: Contents.DataSet.Attributes
Label Path: 'The Contents Procedure'.'IN.BASEBALL'.'Attributes'
--------------
Output Added:  
--------------
Name: EngineHost
Label: Engine/Host Information
Template: Base.Contents.EngineHost
Path: Contents.DataSet.EngineHost
Label Path: 'The Contents Procedure'.'IN.BASEBALL'. Engine/Host Information'
--------------
Output Added:  
--------------
Name: VariablesAlpha
Label: Variables
Template: Base.Contents.Variables
Path: Contents.DataSet.VariablesAlpha
Label Path: 'The Contents Procedure'.'IN.BASEBALL'. Variables'
--------------
Output Added:  
--------------
Name: Sortedby
Label: Sortedby
Template: Base.Contents.Sortedby
Path: Contents.DataSet.Sortedby
Label Path: 'The Contents Procedure'.'IN.BASEBALL'. Sortedby'
--------------
```

The ODS OUTPUT statement can be used to specify one or more output objects from procedures that create multiple output objects. However, SAS does not want to place all of the output objects into a single SAS data set because each object has a different structure. So, we need to create a separate SAS data set for each output object. We can do this using multiple ODS OUTPUT statement or combine the separate output destinations into a single ODS OUTPUT statement. In the code example below, we use the Name value for the output object and a single ODS OUTPUT statement.
Current OUTPUT select list is:
1. Attributes=WORK.CONTENTS1
2. EngineHost=WORK.CONTENTS2
3. VariablesAlpha=WORK.CONTENTS3
4. Sortedby=WORK.CONTENTS4
proc contents data=in.baseball details fmtlen centiles;
run;

Notice in the SAS log each of the four SAS data set created.

OTHER COOL STUFF
Look back at the trace log for the CONTENTS procedure. Remember earlier we noted that the Label value for the output object is the same value that we see in the Results windows? Below is the Results window for the CONTENTS procedure we just ran. Compare this to the trace log.

Have you ever wanted to have some useful information about your SAS data set, such as the number of observations, the observation length, or the creation date?

Well here is some good news. The SAS data set that you can create from the CONTENTS procedure Attributes output object contains some useful information about your SAS data set. At the bottom of the page is what is found for our sample SAS data set.

CONCLUSION
Well, the days of being unable to create a SAS data set from your favorite SAS procedures are gone! The ODS OUTPUT statement now allows you to create a SAS data set from the procedures that are not already supported by the OUT= option.

REFERENCE
SAS version 8.2 on-line documentation “Guide to the SAS Output Delivery System”.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Curtis A. Smith
Defense Contract Audit Agency
P.O. Box 20044
Fountain Valley, CA 92728-0044
Work Phone: 714-896-4277
Fax: 714-896-6915
Email: casmith@mindspring.com