

ODS TAGSETS - a Powerful Reporting Method

Derek Li, Yun Guo, Victor Wu, Xinyu Xu and Crystal Cheng
Covance Pharmaceutical Research and Development (Beijing) Co., Ltd.

Abstract

Understanding some basic knowledge about tagsets can enable you to create those special reports you routinely report every day. Especially when combined tagsets with styles, it can be used to generate good-looking output. In this paper, we take a quick review about the tagsets and show how to apply styles and tagsets together for reporting.

Introduction

It's well known that the Output Delivery System (ODS) used for creating nice-looking report and in pharmaceutical industry it's extensively applied in almost every study as it's the routine work for statistical programmers. But not too many programmers know what is tagsets and some never use tagsets for output. Several papers have introduced the basic grammars, but on introduction level. From our experience, by using style templates and tagsets together, we can create customized output. Further more, we can create our own destination, as shown in NIH format files in our example.

Tagsets is SAS code processed by PROC TEMPLATE procedure. But they are not performing like style templates statements in PROC TEMPLATE. The tagsets code primarily consists of data, events and triggered by events, what is done with the information depends on which event was activated. Through tagsets, we can produce EXCEL, HTML, XML or RTF files, just as we use the styles, but with tagset, it's much more power to control the output that will be produced. In this paper, we will give an introduction on the basics tagset, with styles being combined to show the power of tagsets.

Understanding the tagset

Dozens of tagsets are shipped with SAS. To see the list of tagsets, we can use the following codes to get the SAS-supplied tagsets included in the general template:

```

titled1 *

proc template;
  list tagsets;
run;

```

Obs	Path	Type
1	Tagsets	Dir
2	Tagsets.Accessible	Tagset
3	Tagsets.Cascading_stylesheet	Tagset
4	Tagsets.Chtml	Tagset
5	Tagsets.Colorlatex	Tagset
6	Tagsets.Config_debug	Tagset
7	Tagsets.Csv	Tagset
8	Tagsets.Csvall	Tagset
9	Tagsets.Csvbyline	Tagset
10	Tagsets.Default	Tagset
11	Tagsets.Docbook	Tagset
12	Tagsets.Event_map	Tagset
13	Tagsets.ExcelXP	Tagset
14	Tagsets.Graph	Tagset
15	Tagsets.Graph_rtf	Tagset
16	Tagsets.Html4	Tagset

To see the source codes for a tagset definition, specify the two level name of the tagsets within proc template and the code will then appear in the SAS log window.

```

proc template;
  source tagsets.short_map;
run;

```

```

Log - (Untitled)

proc template;
  source tagsets.short_map;
NOTE: Path 'Tagsets.Short_map' is in: SASHELP.TMPLMST.
define tagset Tagsets.Short_map;
  define event initialize;
  end;
  define event attr_out;
    putq " name=" name;
    putq " value=" value;
  end;
  parent = tagsets.event_map;
end;
run;
NOTE: PROCEDURE TEMPLATE used (Total process time):
      real time           0.20 seconds
      cpu time            0.03 seconds

```

Theoretically, to generate ODS output, SAS procedures use a standard set of events and variables. For example, to generate customized output, SAS creates a customized tagset with customized events. However, in order to customize the events, you need to know the names of the events that ODS uses. A good way to start defining the customized tagset is to use the EVENT_MAP tagset embedded in SAS. This enables you to determine which events are triggered and which variables are used by an event to send output from a SAS process to an output file. When you run a SAS process with TAGSETS.EVENT_MAP, ODS writes XML markup to an output file that shows all event names and variable names as tags. In the output, tag names are the event names. Tag attributes are the variables that have values for those events.

For example, the following statements run ODS MARKUP with TYPE=EVENT_MAP can be used to show which events and variables the ODS are using for various parts of the PROC PRINT output:

```

ods markup type=event_map file='custom-tagset-filename.xml';

proc print data=sashelp.class;
  where Height gt 60;
run;

ods markup close;

```

Except the familiar results in output window, when we open the resulting XML file, part of its content is listed as below:

```

<?xml version="1.0" encoding="windows-1252" ?>
- <doc operator="x2li" sasversion="9.1" saslongversion="9.01.01M3P02022006" date="2012-03-21" time="16:34:04" encoding="windows-1252" event_name="doc" trigger_name="attr_out"
  class="Body" index="IDX" just="c">
- <doc_head event_name="doc_head" trigger_name="attr_out" class="Body" index="IDX" just="c">
  <doc_meta event_name="doc_meta" trigger_name="attr_out" class="Body" index="IDX" just="c" />
  <auth_oper event_name="auth_oper" trigger_name="attr_out" class="Body" index="IDX" just="c" />
  <doc_title event_name="doc_title" trigger_name="attr_out" class="Body" index="IDX" just="c" />
  <stylesheet_link event_name="stylesheet_link" trigger_name="attr_out" index="IDX" just="c" />
- <javascript event_name="javascript" trigger_name="attr_out" class="Body" index="IDX" just="c">
  <startup_function event_name="startup_function" trigger_name="attr_out" class="StartupFunction" index="IDX" just="c" />
  <shutdown_function event_name="shutdown_function" trigger_name="attr_out" class="ShutdownFunction" index="IDX" just="c" />
  </javascript>
</doc_head>
- <doc_body event_name="doc_body" trigger_name="attr_out" class="Body" index="IDX" just="c">
- <proc event_name="proc" trigger_name="attr_out" name="Print" index="IDX" just="c">
  <anchor event_name="anchor" trigger_name="attr_out" class="Body" name="IDX" index="IDX" just="c" />
- <page_setup event_name="page_setup" trigger_name="attr_out" class="Body" index="IDX" just="c">

```

In the XML output generated by EVENT_MAP, PROC PRINT uses events named DOC_HEAD, PROC, TABLE, and so on. The TABLE event uses data from event variables such as STATE, CLASS, and TYPE. After you know the events and variables that generate the output, define the tagset and customize your events. For example, you could redefine the TABLE event to produce customized output.

Now we know a tagset is a type of template that defines how to generate markup language output from a SAS format. You can specify a tagset that create markup language output from ODS. But not like define style in PROC TEMPLATE, during customizing a tagset, we can use SAS DATA step functions, WHERE expressions, macro language statements and so on. And functions can also be used on any statement within the tagset language.

How to use tagsets

1. Learn define styles through tagsets.

At the early days when we want to know how to define a style, we always wonder the relationship between the style and the output, which styles were used in current output, which kind of style attributes were used and so on. Actually, SAS offer a few tagsets to help us to understanding ODS and styles, like STYLE_POPUP, STYLE_DISPLAY, for example, we can use below codes to get the elements definition in our output result.

```

Ods tagsets.style_popup file='example1.html';
Proc print data=sashelp.class;
Run;
Ods tagsets.style_display close;

```

When click the mouse on the resulting HTML file, an option dialog will show like below:

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	
5	Henry	M	14	63.5	
6	James	M	12	57.3	
7	Jane	F	12	59.8	
8	Janet	F	15	62.5	
9	Jeffrey	M	13	62.5	

PROC TEMPLATE Syntax - Windows Internet Exp...

Style Definition

```

STYLE Header /
  FONT_FACE = "Arial, Helvetica, sans-serif"
  FONT_SIZE = 4
  FONT_WEIGHT = bold
  FONT_STYLE = roman
  FOREGROUND = cx0033aa
  BACKGROUND = cxb0b0b0
;

```

Now it is easier to know what kind of style was used on table header, data, system title and so on. In the above example output, there are many other elements that may not show up. To address this problem, we can use STYLE_DISPLAY, that creates sample output for all of the commonly defined elements, this tagset inherits its behavior from the STYLE_POPUP tagset, so it works the same way.

2. Only use tagsets for reporting

As a statistical programmer, one of our big tasks is to create some specific kind of data files for some agency, like FDA and so on. In my experience, one of our client asked us use '\$' as the separator between variables when we transfer the clinical data to them. At the beginning, just as a usual way, we use PUT in DATA STEP to do this work, but we found as the number of variables grown, our codes became complicated, then we begin to think is there any other way to do this task and ODS TAGSETS came to our head. Through use below shot codes, this task was completed very well, no matter how much variables we need handle and we could have more time to drink tea or coffee.

```

proc template;
  define tagset mynih/store=work.template;
  define event data;
    set $data_values[] value;
  end;
  define event row;
    start:
      unset $data_values;
    finish:
      break/if section ne "body";
      eval $count $data_values;
      eval $i 1;
      do/while $i < $count;
        put Strip($data_values[$i])'$'/if not missing($data_values[$i]);
        put '$'/if missing($data_values[$i]);
        eval $i $i+1;
      done;
      put Strip($data_values[$i])/if not missing($data_values[$i]);
      done;
      put NL;
    end;
    stacked_columns = OFF;
  end;
run;

```

3. Using tagsets and styles together for reporting

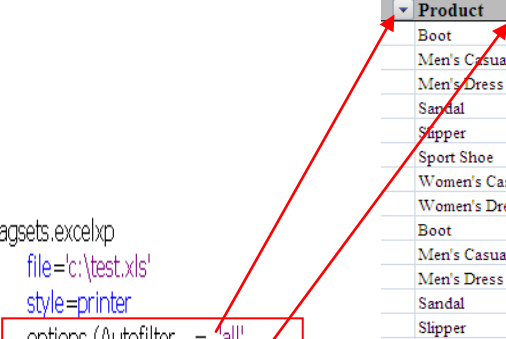
Nobody will challenge that the EXCEL files take a very important role in our work, we prefer use EXCEL to exchange data between different departments, do some simple edit check or calculation in EXCEL, etc. and we also know when we use PROC EXPORT or LIBNAME to create an EXCEL file, the result always looks not good enough, and we have to do some external work to make it look better and easy to understand or easy to handle. But if you know TAGSETS.EXCELXP, which shipped with SAS, you can have much power to control EXCEL output in SAS, below are three examples of use TAGSETS.EXCELXP to create report.

a)

```
ods tagsets.excelxp
  file='c:\test.xls'
  style=printer
  options (Autofilter = 'all'
    Frozen_Headers='1');

proc print data=sashelp.shoes;
run;

ods tagsets.excelxp close;
```



Product	Subsidiary	Stores	Sales
Boot	Addis Ababa	12	\$29,761.00
Men's Casual	Addis Ababa	4	\$67,242.00
Men's Dress	Addis Ababa	7	\$76,793.00
Sandal	Addis Ababa	10	\$62,819.00
Slipper	Addis Ababa	14	\$68,641.00
Sport Shoe	Addis Ababa	4	\$1,690.00
Women's Casual	Addis Ababa	2	\$51,541.00
Women's Dress	Addis Ababa	12	\$108,942.00
Boot	Algiers	21	\$21,297.00
Men's Casual	Algiers	4	\$63,206.00
Men's Dress	Algiers	13	\$123,743.00
Sandal	Algiers	25	\$29,198.00
Slipper	Algiers	17	\$64,891.00
Sport Shoe	Algiers	9	\$2,617.00
Women's Dress	Algiers	12	\$90,648.00
Boot	Cairo	20	\$4,846.00
Men's Casual	Cairo	25	\$360,209.00
Men's Dress	Cairo	5	\$4,051.00
Sandal	Cairo	9	\$10,532.00
Slipper	Cairo	9	\$13,732.00
Sport Shoe	Cairo	3	\$2,259.00
Women's Casual	Cairo	14	\$328,474.00
Women's Dress	Cairo	3	\$14,095.00

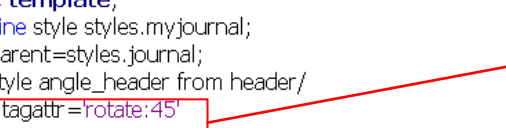
b)

```
proc template;
  define style styles.myjournal;
    parent=styles.journal;
    style angle_header from header/
      tagattr='rotate:45';
  end;
run;

ods tagsets.excelxp file='c:\example.xls' style=myjournal;
proc sort data=sashelp.class out=temp;
  by sex;
run;

proc tabulate data=temp
  order=data missing format=8.0 noseps;
  by sex;
  class age sex name;
  classlev name/s=angle_header;
  var height weight;
  title 'Header 90 rotated';
  tables age,name=' '*(height=' '*median=' '*f=5.3);
run;

ods tagsets.excelxp close;
```



		Alice	Barbara	Carol	Jane	Janet	Joyce	Judy
2								
3	Age	56.5	65.3	.	.	.	.	.
4	13							
5	14	.	.	62.8	.	.	.	64.3
6	12	.	.	.	59.8	.	.	.
7	15	.	.	.	.	62.5	.	.
8	11	.	.	.	.	.	51.3	.

c)

```
ods tagsets.ExcelXP style=sansPrinter file="%outpath./&chkn._&sysdate..xls"
options(Print_Footer='Sheet: &A Page:&P of &N Pages'
       Row_Repeat='header'
       Column_Repeat='header'
       );
```

```
ods tagsets.ExcelXP options(embedded_titles='yes'
       embedded_footnotes='yes'
       absolute_column_width='20'
       sheet_name="%chkn"
       fit2page='yes'
       orientation='landscape'
       gridlines='yes' autofilter_table='2' autofilter='All' sheet_interval='none' frozen_headers='10'
       skip_space='0,0,1,1,1');

```

```
proc report data=tp1 nowd spacing=1 split='|' nothreads style(header)={background=_undef_};
  column aevt deathdt comm rev_com;
run;
```

```
title;
proc report data=out nowd missing headskip headline spacing=1 split='|' nothreads;
  column aevt deathdt comm rev_com;
run;
```

```
ods tagsets.ExcelXP close;
```

Listing: ADVEVES01

More than one Pretreatment/Adverse Event is recorded with Date of Death present.

AEVT	DEATHDT	comm	rev_com
	\$600.00	8	\$600.00 \$200.00
Adverse Term	Death Date	Comments	Reviewer_Comments
Incomplete right bundle branch block		.	
right renal lithiasis		.	
Acute Coronary Insufficiency		.	
Hypokalemia		.	
stomach upset		.	

Conclusion

In this paper, some basic knowledge about ODS TAGSETS was introduced. We also show some advantages of ODS TAGSETS in creating report. ODS TAGSETS give SAS programmer much more power to create report than just use of STYLE. But just as the old saying, everything has two sides, if we want to create a customized tagset, except understand which events and variables will be used, we also need to know the specific tagsets language, like HTML, XML or RTF. Usually this will be the biggest challenge for us, fortunately, SAS allow us customize tagsets base on the shaped tagsets with SAS, this means we can find a existed tagset as the parent of our tagset and only do the needed modifies to get meet our requirement.

REFERENCES

1. Eric Gebhart, “ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets” NESUG 2007
2. Rick Andres, “Printable Spreadsheets Made Easy: Utilizing the SAS Excelxp Tagset” NESUG 2008
3. Eric Gebhart The Beginners Guide to ODS MARKUP: Don’t Panic. Proceedings of the 31-st Annual SAS users Group International Conference.

CONTACT INFORMATION

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

Name: Derek Li, Yun Guo, Victor Wu
Enterprise: Covance Pharmaceutical Research and Development (Beijing) Co., Ltd.
Address: Units1501-1508, Tower A, Gemdale Plaza, No. 91, JianGuo Road
City, State ZIP: Beijing, China 100022
Fax: (86 -10) 5732 3001/3002
E-mail: Derek.li@covance.com
Yun.guo@covance.com
Victor.wu@covance.com
Xinyu.xu@covance.com
Crystal.Cheng@covance.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

Other brand and product names are trademarks of their respective companies.