Paper CT08

Taking better control of ODS outputs using ODS TAGSETS

Kamlesh Patel, Rang Technologies Inc, Piscataway, NJ
Jigar Patel, Rang Technologies Inc, Piscataway, NJ
Vaishali Patel, Rang Technologies Inc, Piscataway, NJ

ABSTRACT

Most of the outputs in pharmaceutical industry are produced by using SAS/ODS. SAS/ODS is powerful to produce various custom reports as per stakeholders' requirements. One of unique feature of SAS/ODS is ODS tagsets. It gives immense power to programmer to control ODS outputs. However, the usage of ODS tagsets among programmer is comparatively less. Hence, this paper would try to cover simple ways to take control of various types of ODS outputs using ODS tagsets.

INTRODUCTION

There are various types of reports format common in pharmaceutical industry. Some of examples are RTF, PDF, Excel, LST, HTML etc. Most of them formats are created by ODS facility in SAS. SAS/ODS is very efficient in giving various option to produce flexible reports. However, SAS/ODS has one of very powerful feature, ODS tagsets. ODS tagsets provide good control over various features of reports.

HOW ODS TAGSET IS DIFFERENT FROM TRADITIONAL ODS

Most of programmer are familiar with the traditional ODS in SAS. Hence, we will see quickly how it ODS tagsets are different than traditional ODS.

Traditional ODS:

- Controls width of columns in output but does not control height of output or cell (so, page breaks are not standard)
- Title/Footnotes goes in title/footnote section of RTF
- Whole table is loaded in memory, so higher memory usage

ODS tagsets (also known as Measured RTF):

- Controls width of columns as well as control height of output or cell (so, page breaks in output can be controlled)
- Title/Footnotes puts in document body
- Consumes only 1 page memory

BACKGROUND OF ODS TAGSETS AND TERMINOLOGY

To start with the understanding of ODS tagsets, please try below two codes.
First code will list all available tagsets. Second code will give source code of tagsets.rtf.

```
*1. To view all tagsets;
proc template;
list tagsets;
run;
*-> Will produce list of tagsets in output window;

*2. To view source code of tagsets;
proc template;
source tagsets.rtf;
run;
*-> Will produce long source code of tagsets.rtf in LOG file.;
```

Below is the snapshot of log file that produce source code.

```
end;

define event changelog;

   trigger changes start;
   putlog "=============================================================================================";
   putlog "History of changes for this tagset";
   putlog $log_note;
   putlog "=============================================================================================";
   iterate $changelog;

   do /while _value_;
      set $ctrl substr(_value_,1,1);

      do /if cmp( _value_, "-");
         putlog "---------------------------------------------------------------------------------------------";

      else /if cmp( $ctrl, ".");
         putlog substr(_value_,2);

      else /if cmp( $ctrl, " ");
         putlog "       " strip(_value_);

      else;
         putlog " * " _value_;
      done;

      next $changelog;
      unset $ctrl;
   done;

   putlog "=============================================================================================";

   trigger changes finish;
end;
```

If you see the source code of tagsets.rtf, it may be overwhelming. But do not worry!!!

You do not have to write that code. But it helps you to understand how ODS tagsets works in backend.
The output is controlled ODS tagsets by sequence of various events and many variables. All those events and variables, you can see in above source code.

To briefly cover the background and terminology related to ODS tagsets, we will quickly see below.
Event

- Occurrence of some specific action that decides what is to be written to the output file.
- Defined by “define event” statement

Variable –

- Like any other SAS variable, it holds data value

Basic information:
There are many tagsets that defines output.

Location of tagsets:	Sashelp.Tmplmst and Sasuser.Templat
Newly created or custom tagsets:	Sasuser.Templat
Referring tagsets:	Two level i.e. tagset.rtf or tagsets.Chtml
SAS Procedure to deal with tagsets:	PROC TEMPLATE

We will now jump to various tips using ODS tagsets for various file formats.

To see how effectively ODS TAGSET can be used, we will see various tips below one by one. -

TIP 1. LEARN TO CREATE SIMPLE TAGSETS TO UNDERSTAND HOW IT WORKS -

We will walk you through creating one simple tagset to understand how ODS tagsets works in backend. We will not see how to create complete tagsets.

STEP 1A:
We will start with very simple code using proc template. Here, we create new tagsets with only 1 event “data”. In that event, we have “PUT” statement with “VALUE” variable.
VALUE Variable > Put the value that is in data.

```
* 1-A > First Tagset Creation;
proc template;
define tagset tagsets.learning;
      define event data; *Start of event;
```

```
            put value; *Putting value;
      end; *End of event;
end;
run;

ods tagsets.learning file="learning.txt";
proc print data=sashelp.class;
run;
ods _all_ close;
```

Output looks like below:

File Edit Format View Help

```
AlfredM1469.0112.5AliceF1356.584.0BarbaraF1365.398.0CarolF1462.8102.5HenryM1463.
5102.5JamesM1257.383.0JaneF1259.884.5JanetF1562.5112.5JeffreyM1362.584.0JohnM125
9.099.5JoyceF1151.350.5JudyF1464.390.0LouiseF1256.377.0MaryF1566.5112.0PhilipM16
72.0150.0RobertM1264.8128.0RonaldM1567.0133.0ThomasM1157.585.0WilliamM1566.5112.
0
```

We see here, all the values from the datasets are put one after another; even those who are in another observations. We don't want such output. But you can connect output with what code we have for ODS tagsets. So, you will understand.
We need to bring records in next line.

STEP 1B:
Now, we will add "nl" options in put statement after value. This option (nl) will take the data value in next line.

```
* 1-B >  First Tagset Creation > Adding nl - New Line;
proc template;
define tagset tagsets.learning;
      define event data;
            put value nl; *nl > Insert new line;
      end;
end;
run;

ods tagsets.learning file="learning.txt";
proc print data=sashelp.class;
run;
ods _all_ close;
```

Below is the snapshot of output. All data are taken to next line.

```
Alfred
M
14
69.0
112.5
Alice
F
13
56.5
84.0
Barbara
F
13
65.3
98.0
Carol
```

STEP 1C:

Now, we will add few more events to the code. In below, we add “header”, “row” events. The “header” event adds the header of datasets which includes name of variable from dataset.

The “row” event adds rows in outputs. We can customize the events. Here, we have added “Start” and “Finish” statement with standard text in put statement. Start work before observation starts and finish work at end of observation.

```
* 1-C > First Tagset Creation > Adding nl - New Line > New events & Text in line >
Row;
proc template;
define tagset tagsets.learning;
       define event data;
              put value nl;
       end;
       define event header;
              put  value nl;
       end;
       define event row;
       start:
              put "----New Record Begins Here----" nl;
       finish:
              put "----Record Ends Here----" nl;
       end;
end;
run;

title "Class";
ods tagsets.learning file="learning.txt";
proc print data=sashelp.class;
run;
ods _all_ close;
```

Please see below output snapshot-

```
----New Record Begins Here----
Obs
Name
Sex
Age
Height
Weight
----Record Ends Here----
----New Record Begins Here----
1
Alfred
M
14
69.0
112.5
----Record Ends Here----
----New Record Begins Here----
2
Alice
```

In above output, we can see, before beginning of each record, there is line and at the end of record also there is a line.

STEP 1D:

We have added “system_title” event for title with start and finish statement to create dash line before and after title.

```
* 1-D - Adding nl > New Line > Text in line > Row;
proc template;
define tagset tagsets.learning;
        define event header;
                put  value nl;
        end;
        define event data;
                put value nl;
        end;
        define event row;
        start:
                put "=========" nl;
        finish:
                put "=========" nl;
        end;
        define event system_title;
        start:
                put "-------------------------------------------" nl;
                put Value nl;
        finish:
                put "-------------------------------------------" nl;
        end;
end;
run;

title "Title 1: Class Data";
ods tagsets.learning file="learning.txt";
proc print data=sashelp.class;
run;
ods _all_ close;
```

Please see below output snapshot –

```
|-----------------------------------------
Class
-----------------------------------------
=========
Obs
Name
Sex
Age
Height
Weight
=========
=========
1
Alfred
M
14
69.0
112.5
=========
```

Here, we have customized title. We have added dashed line before and after title.

STEP 1E:

Now, we will try to create output in table form by displaying columns one by another. In the output, to put data columns for one after another with horizontal tab as separator, we can try this code.

```
put VALUE "   " / if cmp( COLSTART , "1" );
```

This one asks SAS to put value of data and tab when COLSTART (i.e. one of the variable who hold column number) value is respective columns. In bracket, there is horizontal tab.

We want records to start in next line so "nl" in last PUT statement.

This code, `"if cmp( COLSTART , "1" )"`, is IF Condition in ODS tagsets. CMP compares value of COLSTART and "1". If it is true, action item before slash will be executed.

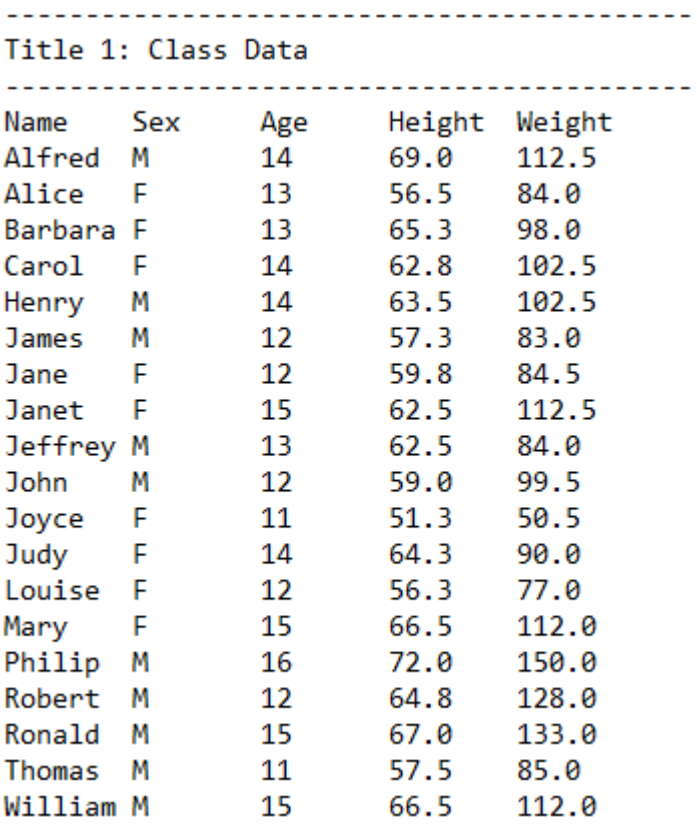
Note that we have removed row event to keep simple.

```
* 1-E - Adding nl > New Line > Text in line > Row > Creating columns;
proc template;
      define tagset tagsets.learning;
            define event header;
                  put VALUE "  " / if cmp( COLSTART , "1" );
                  put VALUE "  " / if cmp( COLSTART , "2" );
                  put VALUE "  " / if cmp( COLSTART , "3" );
                  put VALUE "  "/ if cmp( COLSTART , "4" );
                  put VALUE "  " nl/ if cmp( COLSTART , "5" ) ;
            end;

            define event data  ;
                  put VALUE "  " / if cmp( COLSTART , "1" );
                  put VALUE "  " / if cmp( COLSTART , "2" );
                  put VALUE "  " / if cmp( COLSTART , "3" );
                  put VALUE "  "/ if cmp( COLSTART , "4" );
                  put VALUE "  " nl/ if cmp( COLSTART , "5" ) ;
            end;
            define event system_title;
            start:
                  put "-----------------------------------------" nl;
                  put Value nl;
            finish:
                  put "-----------------------------------------" nl;
            end;
```

```
      end;
run;

title "Title 1: Class Data";
ods tagsets.learning file="learning.txt";
proc print data=sashelp.class noobs;
run;
ods _all_ close;
```

With above code, we can get below output.

Snapshot of output.



Title 1: Class Data

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84.0
John	M	12	59.0	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90.0
Louise	F	12	56.3	77.0
Mary	F	15	66.5	112.0
Philip	M	16	72.0	150.0
Robert	M	12	64.8	128.0
Ronald	M	15	67.0	133.0
Thomas	M	11	57.5	85.0
William	M	15	66.5	112.0

TIP 2. IDENTIFY EVENT OF INTEREST -

As we show before, the event is important unit of any tagset code to produce the output. So, if you have to identify some specific event of interest, it may look difficult. However, there is a simple way to know about which events are playing role in particular code. You can use EVENT_MAP to identify events that are being used by output. I recommend using two specifically, EVENT_MAP and TEXT_MAP.

In below code, you can see how event maps files can be applied. This can generate two files in XML.

```
ods listing;
ods markup type=Text_Map    file='C:\Users\Text_Map.xml';
ods markup type=EVENT_Map    file='C:\Users\Event_Map.xml';

proc print data=sashelp.class;
run;

ods markup close;
ods listing close;
```

This creates event map in XML files. One you open, you can review and identify with keywords. If EVENT_MAP output is loaded with many information, you can take its shorter version of map i.e. short_map output. The text_map file also helps to identify events of interest in the output.

In the snapshot below (event_map.xml file), you can see, there are various events shown (highlighted in yellow). E.g. table_head, row, header, table_body, row, data etc. All these events are triggering and playing role in developing this outputs.

Same events can be found in text_map file as well (not shown below).
Snapshot of Event_Map File.

```
- <table_head just="c" trigger_name="attr_out" event_name="table_head" index="IDX" output_label="Data Set SASHELP.CLASS" output_name="Print"
  colcount="1">
    - <row just="c" trigger_name="attr_out" event_name="row" index="IDX" output_label="Data Set SASHELP.CLASS" output_name="Print" colcount="1"
      section="head">
        <header just="r" class="header" trigger_name="attr_out" event_name="header" name="Obs" index="IDX" label="Obs" value="Obs"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="3" col="1" col_id="1" section="head"
          data_row="0" row="1"> </header>
        <header just="l" class="header" trigger_name="attr_out" event_name="header" name="Name" index="IDX" label="Name" value="Name"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="7" col="2" col_id="2" section="head"
          data_row="0" row="1"> </header>
        <header just="l" class="header" trigger_name="attr_out" event_name="header" name="Sex" index="IDX" label="Sex" value="Sex"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="4" col="3" col_id="3" section="head"
          data_row="0" row="1"> </header>
        <header just="r" class="header" trigger_name="attr_out" event_name="header" name="Age" index="IDX" label="Age" value="Age"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="4" col="4" col_id="4" section="head"
          data_row="0" row="1"> </header>
        <header just="r" class="header" trigger_name="attr_out" event_name="header" name="Height" index="IDX" label="Height" value="Height"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="7" col="5" col_id="5" section="head"
          data_row="0" row="1"> </header>
        <header just="r" class="header" trigger_name="attr_out" event_name="header" name="Weight" index="IDX" label="Weight" value="Weight"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="7" col="6" col_id="6" section="head"
          data_row="0" row="1"> </header>
    </row>
</table_head>
- <table_body just="c" trigger_name="attr_out" event_name="table_body" index="IDX" output_label="Data Set SASHELP.CLASS" output_name="Print"
  colcount="1">
    - <row just="c" trigger_name="attr_out" event_name="row" index="IDX" output_label="Data Set SASHELP.CLASS" output_name="Print" colcount="1"
      section="body">
        <header just="r" class="rowheader" trigger_name="attr_out" event_name="header" name="Obs" index="IDX" label="Obs" value="1"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="double" colcount="1" colwidth="8" col="1" precision="0" scale="0"
          col_id="1" section="body" data_row="1" row="2" rawvalue="P/AAAAAAAAA="> </header>
        <data just="l" class="data" trigger_name="attr_out" event_name="data" name="Name" index="IDX" label="Name" value="Alfred"
          output_label="Data Set SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="8" col="2" precision="0" scale="0"
          col_id="2" section="body" data_row="1" row="2"> </data>
        <data just="l" class="data" trigger_name="attr_out" event_name="data" name="Sex" index="IDX" label="Sex" value="M" output_label="Data Set
          SASHELP.CLASS" output_name="Print" type="string" colcount="1" colwidth="1" col="3" precision="0" scale="0" col_id="3" section="body"
```

There are many other types of markup available in tagsets that can help to understand code.

TIP 3. HOW TO KNOW MORE OPTIONS IN CURRENT TAGSETS -

For various ODS tagsets, SAS is equipped with multiple options. Those options make it easy to customize the output. Availability of option depends on tagsets. To see various options available to use in tagset, there is simple way.

```
* Identity options available in ODS tagsets ;
ods tagsets.rtf file='test1.html' options (doc="help");
proc print data=Sashelp.Class;
run;
ods _all_ close;
```

In above code, “options (doc="help")” list out in log file various options available to customize the output. output all options in LOG.

```
================================================================================
The RTF Tagset Help Text.

This Tagset/Destination helps with the creation of RTF files
for MS Word

================================================================================

These are the options currently supported by this tagset.

Sample usage:

ODS TAGSETS.RTF OPTIONS(doc='Quick',
                        CONTENTS='yes');

ODS TAGSETS.RTF OPTIONS(SPECIFIC_OPTION='value');


Debug_Level: No default value.
     Current value: 0
     Usage: OPTIONS(Debug_Level=1)
     Description:
       Determine what debugging information should be printed
       to the log.  The values expected are numeric and can be used to
       take whatever action is needed. Used in tagsets being debugged,
       but requires a local tagset to be modified.

Doc:  No default value.
     Help: Displays introductory text and options.
     Quick: Displays available options.
     Settings: Displays Current settings.
```

As we see above in image, there are many options available for tagset.RTF.
We will go over few options –

1. TROWD: Inserts raw RTF specifications directly onto the table row header descriptions.

2. TRHDR: Inserts raw tablerow RTF specifications directly onto the table row header descriptions.

3. TROWHDRCELL: Inserts raw text on the trow cells. (Header Cell value).

The string must be rtf_control_string. This specifies RTF control words that customize output.

Here, you can insert string that controls RTF output. Please remember, it is not any string. Here, you have to refer RTF Specifications to know what changes you want to achieve in output. Accordingly, you can put string in header.
E.g. trrh900 > number represent height of cell.
Similarly, we can use other options like TRHDR, TROWHDRCELL to control the cell and header.

For using right string, you can refer Rich Text Format (RTF) Specification, version 1.6. With referring that document and with above 3 options, you have great flexibility.

4. VSPACE: uses the PARSKIP event to place a space table in the document.

This option will insert/remove line after title and before table depending on VSPACE value is yes or no.

There are many other options that can be used in the ODS tagsets. However, one can refer those and apply those.

CONCLUSION

ODS tagsets is equipped to provide user extensive control over various types of output file formats like RTF, HTML, EXCEL etc. Even with introductory knowledge of ODS tagsets, a SAS Programmer can make effective use of functionality to make powerful custom reports in various formats. New learning of event can be expanded by practicing and deep diving into this area. However, this new learning with tips available can set a new path for gaining expertise in ODS tagsets. Even without customizing ODS tagsets, a SAS Programmer with available options can greatly take control of output.

REFERENCES (HEADER 1)

1. https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.5&docsetId=odsproc&docsetTarget=n1pdo7tyve7zgkn1p7hba5s92rs0.htm&locale=en
2. https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/014-30.pdf
3. https://support.sas.com/resources/papers/proceedings/proceedings/sugi27/p178-27.pdf
4. RTF Specification: http://docshare01.docshare.tips/files/22211/222118993.pdf

~~RECOMMENDED READING (HEADER 1)~~

~~Recommended reading lists go after your acknowledgments. This section is not required.~~

CONTACT INFORMATION

(In case a reader wants to get in touch with you, please put your contact information at the end of the paper.)
Your comments and questions are valued and encouraged. Contact the author at:

Kamlesh Patel
Rang Technologies Inc
Piscataway, NJ, 08854
Email: kamlesh.patel1@rangtech.com
Web: www.rangtech.com