

An Introduction to ODS Tagsets

**Kevin Davidson
FSD Data Services, Inc.**

ABSTRACT

Beginning in SAS Version 8.2, ODS MARKUP allows users to create customized output using tagsets. There are over 20 different tagsets that ship with Version 8.2. This paper will review the steps needed to modify a SAS shipped tagset to achieve a desired customized look for HTML output.

INTRODUCTION

ODS Markup allows you to customize your SAS output to precisely the way you want it. You have a choice of using one of the standard markup tagsets that ship with SAS software or you can customize your own. This paper serves as an introduction to tagsets and shows how you can make a modification once and have it persist so the output comes out the way you want it.

What are tagsets?

ODS Markup generates markup language output from tagsets shipped with SAS or user defined tagsets created with Proc Template (on no!!). I'll walk you through an example of creating the Proc Template code that will hopefully make that task seem a little less daunting. Apparently the impetus to create tagsets came from the fact that there are so many flavors of XML that it was realized that it would be impossible for SAS developers to keep up with the demand for the various implementations. Thus, MARKUP was introduced so that end-users could create their own version(s).

What is the difference between creating a Table template, a Style template, and ODS Markup?

Here is where you might use each of these three:

Table template – most procedures (Print, Report, Tabulate, and Freq being the exceptions) have table templates. You modify one of these when you want to alter how the output from the procedure appears (formats, headers, order of columns, table footers, etc.). Independent of destination. In Jeopardy speak, 'Table Template' is the answer to the question "What do I need to change to modify the default layout of output from my procedures"?

Style template – used to alter the look of a given output document and is dependent on the destination. This is where you alter font size, color, weight, etc. Jeopardy speak – "What do I need to change to modify the default appearance of my output"?

ODS MARKUP Tagsets— use this when you want to create markup language output (either HTML, XML, CSV, or some other markup variant). Jeopardy speak (simplified) – “What do I need to change to change the way my HTML tags are created”?

What groups of tagsets exist?

You can view a list of the tagsets that ship with SAS by running the following code:

```
proc template;
  list tagsets;
run;
```

Here is at least a partial listing of the Tagsets that are currently available from SAS:

8.2 Tagsets	
CHTML	Produces a compact, minimal HTML that does not use style information. It does produce a hierarchical table of contents.
COLORLATEX	Produces color LaTeX, which is a document preparation system for high-quality typesetting. It also generates a stylesheet. The output can be rendered to PDF.
CSV	Produces comma separated values output, which contains columns of data that are separated by commas. CSV produces the tabular output only.
CSVALL	Produces comma separated values output, which contains columns of data that are separated by commas. CSVALL produces the tabular output with titles, notes, and bylines.
CSVBYLINE	Produces comma separated values output, which contains columns of data that are separated by commas. CSVBYLINE includes bylines.
DEFAULT	Is the default value and produces XML markup, referred to as ODSXML.
DOCBOOK	Produces XML output that conforms to the DocBook DTD by OASIS.
EVENT_MAP	Creates XML output that shows which events are being 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 EVENT_MAP, ODS writes XML markup to an output file that shows all event names and variable names as tags. The output helps you to create your own tagsets.
GRAPH	To be used for SAS/GRAPH output in a later release.
GTABLEAPPLET	Produces XML output for use with the SAS/GRAPH GTableApplet.
HTML4	To be used for embedded stylesheets. HTML4 will replace ODS HTML in Version 10.

HTMLCSS	Produces HTML that is similar to ODS HTML with cascading stylesheets.
IMODE	Produces HTML that is basically a column of output, separated by lines. This tagset is used by the Japanese telephone service provider (NTT).
LATEX	Produces LaTeX, which is a document preparation system for high-quality typesetting. It also generates a stylesheet. The output can be rendered to PDF.
LATEX2	Produces LaTeX, which is a document preparation system for high-quality typesetting.
NAMEDHTML	Creates HTML output like STYLE_POPUP but with all the objects labeled similar to ODS TRACE. The output helps you to understand ODS and styles.
ODSSTYLE	Creates PROC TEMPLATE code for STYLESHEET= file specification. The output helps you to create and modify styles.
PHTML	Produces a basic HTML that uses twelve style elements and no class attributes.
PYX	Produces PYX, which is a simple, line-oriented notation used by Pyxie to describe the information communicated by an XML parser to an XML application. Pyxie is an Open Source library for processing XML with the Python programming language.
SASIOXML	Produces generic XML markup.
SASXML	Produces generic XML markup.
SASXMOG	Produces XML markup that is similar to the Oracle8iXML implementation used by ORACLE but is more generic. (Note that this is the tagset used by the SAS XML LIBNAME engine for XMLTYPE=GENERIC.)
SASXMOH	Produces very simple HTML markup. (Note that this is the tagset used by the SAS XML LIBNAME engine for XMLTYPE=HTML.)
SASXMOIM	Produces XML markup that is supported by the Open Information Model (Database Schema Model) proposed by the Metadata Coalition (MDC) as vendor- and technology-independent, conforming to the 1.0 specification. (Note that this is the tagset used by the SAS XML LIBNAME engine for XMLTYPE=OIMDBM.)
SASXMOR	Produces XML markup that is equivalent to the Oracle8iXML implementation, which is used by ORACLE. (Note that this is the tagset used by the SAS XML LIBNAME engine for XMLTYPE=ORACLE.)
SHORT_MAP	Creates a subset of the XML output that is created by EVENT_MAP.
STYLE_DISPLAY	Creates HTML like STYLE_POPUP but produces a sample page of output for you to click. The output helps you to create

	and modify styles.
STYLE_POPUP	Creates HTML like HTMLCSS, but if you're using Internet Explorer, STYLE_POPUP displays a window that shows the resolved ODS style definition for any item that you click. The output helps you to understand ODS and styles.
TEXT_MAP	Creates text output as an alternative to EVENT_MAP that shows which events are being triggered as ODS handles the output objects.
TPL_STYLE_LIST	Creates HTML output similar to EVENT_MAP but lists only a subset of the possible attributes. The information is in a bulleted list. The output helps you to understand tagsets and styles.
TPL_STYLE_MAP	Creates XML output similar to EVENT_MAP but lists only a subset of the possible attributes. The output helps you to understand tagsets and styles.
TROFF	Produces Troff markup, which is a text-formatting programming language for high-quality photo-typesetters and laser printers
WML	Uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a simple href list for a table of contents.
WMLOLIST	Uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with an option list for the table of contents.

Note that you can download the most recent 8.2 tagsets from:

http://www.sas.com/rnd/base/topics/odsmarkup/tagsets_template_8.sas

CUSTOMIZING TAGSETS

Let's start digging into an example that shows you how you customize your own tagset and why that might be helpful.

First, there are a couple of things to understand about tagsets, namely events and statements. Let's explore these things together by looking at some actual tagset code. We will use the CHTML tagset that was chosen because it creates a file with compact, minimal HTML code (and we want to keep this example simple):

For discussion purposes, let's create a very simple HTML page with the following program:

```
data a1;
  x=3; y=4; output;
  x=1; y=10; output;
```

```

run;

ods markup tagset=tagsets.chtml body='c:\era.html';
title;
proc print data=a1;
run;
ods markup close;

```

Results in a page that looks like the following.

Obs	x	y
1	3	4
2	1	10

The html source code produced looks like the following:

```

<html>
<p>
<table columns="3" cellpadding=2 border=1>
<a name="IDX"><thead>
<tr><td align="center">Obs</td><td align="center">x</td>
<td align="center">y</td></tr>
</thead>
<tbody>
<tr><td align="right">1</td><td align="right">3</td><td align="right">4</td></tr>
<tr><td align="right">2</td><td align="right">1</td><td align="right">8</td></tr>
</tbody>
</table></p>
</html>

```

Suppose we decide we want to make a couple of permanent changes to the output produced. Namely:

- 1) Substitute the heading of 'Record #' in place of 'Obs' because most viewers of our report will not be SAS users.
- 2) Flag any values that have a value of 10 (let's do something simple like put an asterisk next to it).

We know what changes we want to make, now how do we do it?

First, we need a way of changing the tagset code to our own customized copy. Fortunately, there is an easy way to do that.

```

proc template;
source tagsets.chtml;
run;

```

Running the program listed above creates about 150 lines of code. That code is the entire set of Proc Template code that runs when the tagset is utilized. Thus, somewhere in the code should be the sections that we need to change to enact our desired changes. In looking at the HTML source code, it looks like sections which produce '<td>' tags is what we want to find to change the 'obs' reference and to highlight values of 10. After looking through the Proc Template code, here is the likely candidate for change:

```
define event data;
  start:
    put "<td";
    trigger rowcol;
    putq " align=""center"" / if cmp( "c" , JUST );
    putq " align=""right"" / if cmp( "r" , JUST );
    putq " align=""right"" / if cmp( "d" , JUST );
    put ">";
    put VALUE;
  finish:
    put "</td>";
end;
```

At first glance, the code should look both familiar and new. Here are some key concepts and statements:

- 1) 'define event data' – when a tagset is called upon, a sequence of 'events' happen. These events are what is captured in a given tagset. The CHTML tagset has 27 different events defined with names that generally suggest their function, such as 'table_head', 'breakline', and 'row'. It is not necessary to have a mastery of all these events in order to customize your own tagset. You simply need to be able to pick out the events you need to modify!
- 2) 'trigger' – does what it sounds like it does, it 'triggers' another event to fire.
- 3) 'Putq' – this is one of the statement that you can use to write out your markup code. 'Putq' places quotes around variable names, 'putl' adds a new line, whereas 'put' functions like the 'put' statement you are used to. There are also indentation statements.
- 4) 'if cmp("d" , JUST)' – this is the tagsets form of a conditional if statement. The 'if' part comes after the slash. 'JUST' and 'VALUE' are automatic variables, of which there are quite a few (see ODS styles).
- 5) 'start' and 'finish' – these are labeled sections within an event that signal for sections of code to run at the beginning or end of an event.

For a more complete explanation of events and the coding of events, see the paper by Eric Gebhart (listed in the reference section).

We have identified the section of code we need to change, what next? Because we are going to use the functionality of the CHTML tagset with just a few modifications, we have two choices:

- 1) submit the template code in its entirety (~150 lines) with the changes embedded among the other code that we are not going to alter
- 2) submit only the changes with a reference to use CHTML as the parent.

For brevity of coding, let's go with option #2.

Here is our Proc Template code to create a new tagset, which we will call 'MYhtml'.

```
proc template;
  define tagset tagsets.MYhtml;
    parent=tagsets.html;
    define event data;
      start:
        put "<td";
        trigger rowcol;
        putq " align=""center"" / if cmp( "c" , JUST );
        putq " align=""right"" / if cmp( "r" , JUST );
        putq " align=""right"" / if cmp( "d" , JUST );
        put ">";
        put "Record #" / if cmp("Obs",VALUE); *#1;
        break / if cmp("Obs",VALUE); *#2;
        put "*" /if cmp("10",VALUE); *#3;
        put VALUE;
      finish:
        put "</td>";
    end;
  end;
run;
```

Within the 'define event data' block, the only change is that the lines followed with "*#x" have been added. The logic within those lines say:

Line #1 – if the automatic VALUE variable='Obs' then write 'Record #' instead

Line #2 – if the automatic VALUE variable='Obs' then don't process any more statements

Line #3 – if the automatic VALUE variable has a value of 10, write an asterisk

Once the new tagset is created, then running the same ODS and Proc Print statements as above with a reference to 'MYHTML' rather than 'CHTML' creates the following HTML page:

Record #	x	y
1	3	4
2	1	*10

CONCLUSION

Tagsets are a very powerful tool for creating HTML and other markup language code the way you want it. Custom tagsets can be created and stored in user-defined libraries and are easily accessed via the ODS statement. With a fairly short time investment, you can be creating and using your own tagsets.

REFERENCES

Hamilton, Jack, "An Application of ODS Tagsets", SUGI 27, First Health, West Sacramento, CA.

Gebhart, Eric, "MARKUP: The Power of Choice and Change", SUGI 27, SAS Institute Inc., Cary, NC.

CONTACT INFORMATION

Kevin Davidson, Ph.D.
FSD Data Services, Inc.
2020 Southwest Freeway, Suite 206
Houston, Texas 77098
713-942-8436
kevind@fsd.nu
www.fsd.nu

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. ® indicates USA registration.