An Application of ODS Tagset to Produce HTML Files for Clinical Study Reports

Li Luo, Wyeth Pharmaceuticals, Collegeville, PA

ABSTRACT

This paper introduces the concepts necessary to understand and apply the advanced features of the SAS Output Delivery System (ODS). An approach to generate HTML files with a powerful ODS HTML tagset will be discussed. The focus is on the creation of a department defined standard ODS tagset with PROC TEMPLATE and applications of this tagset to create Submission Ready Components Standard compliant documents for Clinical Study Reports. Code capabilities are reviewed and different example invocations are shown.

INTRODUCTION

In the pharmaceutical industry, it is quite desirable for rich text format (RTF) or word tables to be generated automatically by SAS programs and later inserted directly into a word document for a final Clinical Study Report. For this reason, many tools, macros and templates have been developed over the past several years. SAS’s Output Delivery System (ODS) feature enables the creation of various new file types including Rich Text Format (RTF), PostScript and HTML. The standard default style used to control the output is generally well organized and sufficient for most purposes. By using style and table templates, programmers have greater control on report details such as colors, fonts, size, data justification, order and labels to meet the company standards. This paper will show how an ODS tagset was developed to generate company standard output including HTML.

REVIEWING ODS MARKUP BASICS

The MARKUP destination is the most powerful Output Delivery System (ODS) destination. ODS MARKUP is an addition to the Output Delivery System starting with SAS version 8.2. Markup languages produce SAS output that is formatted using one of many different markup languages such as Extensible Markup Language (XML), HTML, CSV (comma separated values), WML (Wireless Markup Language), CHTML (compact HTML), and many others that you can access with a Web browser. A markup language is a textual way of adding context or formatting to data. A markup language can be as simple as CSV in which the only markup consists of commas that separate the data values. The most well-known markup language is HTML. The actual HTML consists of the tags that are enclosed by angle brackets (<, >) and are placed around the content. These tags provide structure, meta and formatting information for the contents. ODS MARKUP relies on programs that contain instructions for adding these markup statements to the output. These programs are called tagsets.

UNDERSTANDING TAGSET

MARKUP generates markup language output from tagsets shipped with SAS or user defined tagsets created with PROC TEMPLATE. Tagsets are templates, as in templates for PROC TEMPLATE. For SAS 9.1, use PROC TEMPLATE to get a full list of the tagsets. Any of those tagsets can be used with MARKUP destination just by specifying the tagset in the ODS MARKUP statement.

Basic Syntax:

```
ods markup file=    tagset (or type=) option allows you to specify the type of markup language to be used with your SAS output. Without a tagset (or type=) option, the default output is an XML document. We specify some tagset keywords as ODS destinations. The tagset determines the type of markup that we will have in our output file.

ods markup body='class.html' tagset=phtml;
more SAS code...
```
There is a tagset named cHTML, which stands for Compact HTML. cHTML is a subset of HTML and has minimal stylistic controls. cHTML has just the basics of HTML and was originally intended for phones and PDAs. Its compact size and simplicity made it perfect for small devices. Because of its simplicity, cHTML is popular among those who want black-and-white pages, and simple HTML. cHTML is also a good starting point for creating a custom-made HTML tagset of your own. To see the source for a tagset definition, use PROC TEMPLATE and specify the two-level name of the tagset. For example, to see the source of a SAS tagset that generates HTML output:

```
proc template;
  source tagsets.chtml;
```

The source for tagset.chtml consists of:

- A DEFINE TAGSET statement that names the tagset definition.
- Event definitions that define what is written to the output file.
- Tagset definition attributes that set the output type, the character to use for line breaks, and so on

## Creating a Tagset

There are at least two ways to create a tagset. You can modify an existing tagset, or you can start from scratch. We created a custom HTML tagset by modifying existing tagset which inherits most of its properties using the PARENT=attribute in the DEFINE tagset statement. We need minimal style so that the table can be inserted into a CSR easily. cHTML is our choice, which can produce a compact, minimal HTML that does not use style information. The following code defines a new tagset named TAGSETS.MYTAGS that creates customized HTML output. Most of the required formatting is available in the SAS tagset TAGSETS.CHTML, so the PARENT= attribute is used for the new tagset TAGSETS.MYTAGS to inherit events from TAGSETS.CHTML. Note that the ODS PATH statement is specified at the beginning to establish the search list.

```
ods path sasuser.templat (update)
  sashelp.tmplmst (read);
proc template;
  define tagset tagsets.mytags /store=sasuser.templat;
    parent=tagsets.chtml;
  define event colspecs;
    put "These are my new colspeccs" nl;
  end;
```

## Events and Statements

The most important concept in tagsets is the event. An event occurs when ODS begins a specific phase of the output processing. Statements are associated with events, and tell ODS what to do when an event is encountered. The most important statements for simple tagets are put, putl, and putq, which write to the ODS output file. The set and unset statements create and remove memory variables. Some events contain other events. For example the document event is the first event triggered by ODS. All other events occur after the document start event and before the document finish event. Similarly there are events, such as table row and data events, which occur in between the start and finish event of the table. Here are some key points about events:

- An event is an action that can be used to generate output. Events are usually initiated by SAS but can also be triggered by other events.
- The DEFINE EVENT statement assigns a name to an event definition.
- An event definition can include start and/or finish sections that specify different actions. If the event definition does not include either a start or finish section, the event is stateless, which means that no matter how the event is called, all of the actions in the event are executed.
• An event definition can execute another event using the TRIGGER statement. If you are in the start section of an event, then any event triggered will also run its start section. If you are in the finish section, then the triggered event will run its finish section. If a triggered event does not have start or finish sections, then it will run the statements it does have. A trigger can also explicitly ask for an event’s specific section.
• Events can perform actions based on conditions.

Start: is what defines the Start portion of an event.
Finish: is what defines the Finish portion of an event.

Here is what a very simple HTML table event might look like

Define event table;
    start:
        put "<table>" nl;
    finish
        put "</table>" nl;
end;

The ndent and xdent statements are used to control indentation in the output. An event can have a statement condition, which controls whether the statement will be executed; it’s similar in purpose to an IF statement.

VARIABLES

Variables are symbolic references to values. In simple tagsets, variable will have been set by ODS supervisor. There are dozens of variables, which might be set automatically; they are listed in the tagset documentation. These include the ODS style variables. Then there are the variables that hold the basic data for the event. There is also another kind of variable, dynamic variables. If you are familiar with proc template you may already be familiar with dynamic variables. Dynamic variables can be listed with a dynamic statement. Or they can be referenced by preceding them with an @. These variables are dynamic in that they are not defined within the confines of ODS. But are defined by our internal users. The biggest creator of dynamic variables is SAS Graph. Finally there are user-defined variables. You can assign a string or the value of another variable to them. These variables are designated with a $.

TAGSET DESIGN AND TEMPLATE SYNTAX

In 2005, a custom-made HTML tagset called ldtagset was developed by Michael Jessup at Wyeth to generate HTML tables. It has been used since then with minimal upgrades. The tagset is very powerful and flexible working on SAS version 9 or later. When the tagset is loaded, it will create a definition called html.standard. This tagset is for formatting output in a word-ready table. It will generate HTML output to be used as in-text tables as based upon the Submission Ready Components Standard for Clinical Study Reports. All titles will be placed in a single cell at the beginning of the HTML table. All footers will be placed in a single cell at the end of the table. Titles are centered and footnotes left justified. All text will be displayed in Time New Roman and all column headers, which span more than one column, will be placed in an outlined cell. Titles and footnotes are in the same table as the data. The generated files should be viewed in Microsoft Word 2000 or later version. Using Internet Explorer or Microsoft Excel is OK but because of differences in the rendering engines the output may appear differently than in it will in Microsoft Word.

Some examples of the template syntax:

PART 1 – STORE THE TEMPLATE
Define the template and store in the working directory

ods path (append) WORK.html(update);
proc template;
define tagset html.standard /STORE=work.html;
parent=tagsets.chtml;

The Append sub-option to the ODS PATH statement allows us to add the path as the last path on the search list. The Pre-append option allows us to append this on to the beginning of the search list. This prevents from having to redefine the search list.

/*WE REMOVE THE SUBSTITUTIONS TO ALLOW EMBEDDED HTML WITHIN THE DATA CONTENTS*/

map='';
mapsub='';

mvar sysdate;
mvar systime;
mvar reportno;
mvar __footnoteTarget;

PART 2 – TITLES AND FOOTNOTES

/*WHEN THE TITLE GROUP BEGINS WE DEFINE A VARIABLE TO COLLECT ALL TITLES
UNTIL WE ARE READY TO OUTPUT THE IN THE FIRST TABLE CELL*/

define event system_title_group;
    unset $titles;
    set $titles "";
end;

/*WHEN THE FOOTER GROUP BEGINS WE DEFINE A VARIABLE TO COLLECT ALL FOOTERS SO
THEY CAN BE OUTPUT TOGETHER IN THE LAST CELL OF THE TABLE*/

define event system_footer_group;
    start:
        unset $footers;
        set $footers "";

The following create output footnote in the last cell of the table with a border at the top and finally close the table

finish:
    put "<tr><td colspan=""$COLCOUNT"" align=""left"" style=""border-top-width: 1px;border-top-style: solid;border-top-color: #000000"">" NL;
    put "<a name=""__footnoteTarget"">" /when exists(__footnoteTarget);
    put "<p style=""font-size: 10pt; font-family: Times New Roman;font-weight: normal;"">"<nobr>" NL;
    put $footers "</nobr><br>
    put reportno " - " /when exists(reportno);
    put sysdate " " systime "</p>";
    put "</a>" /when exists(__footnoteTarget);
    put "</td></tr>" NL;
    put "</table>" NL;
unset $TABLE_OPEN;
end;

PART 3 TABLES

When we start a new table we first check to see if a table is already open, and if so close it, we then open a new table and note that a new table has been opened.

define event table;
    start:
        putq "</table>" NL /when exists($TABLE_OPEN);
        set $TABLE_OPEN "Y";
        put "<table border="0" cellspacing="5" cellpadding="0">" NL;
    end;

The following code is same as parent tagset except we add a style to output the text in the correct font and size.

define event data;
    start:
        put "<td style="font-size: 10pt; font-family: Times New Roman;">"
        trigger rowcol;
        putq " align="center"" /when cmp("c", JUST);
        putq " align="right"" /when cmp("r", JUST);
        putq " align="right"" /when cmp("d", JUST);
        put " nowrap" /if no_wrap;
        put "">";
        put "<pre>" /if asis;
        trigger hyperlink /if exists( URL);
        put VALUE /if ^exists(URL);
    finish:
        trigger hyperlink /if exists( URL);
        put "</pre>" /if asis;
        put "</td>" NL;
    end;

PART 4 HEADER

define event header;
    start:
        /*AS HEADERS ARE OUTPUT WE INTRODUCE THE STANDARD TAGGING AS WELL AS BORDERS FOR CELLS WHICH SPAN MULTIPLE COLUMNS*/
        put "<th ";
        trigger rowcol;
        do /if cmp("c", JUST); put " align="center"";
        else /if cmp("r", JUST); put " align="right"";
        else /if cmp("d", JUST); put " align="right"";
        done;
        put " nowrap" /if no_wrap;
        put " valign="bottom";
        put " style="font-size: 10pt; font-family: Times New Roman; font-weight: normal;";
EXAMPLE OF APPLICATIONS AND OUTPUT

Example 1: HTML special characters and HTML code

Care must be taken when using the special character in output. For more information on HTML the official HTML4 specification is available at [http://www.w3.org/TR/html4](http://www.w3.org/TR/html4). There is also a tutorial available at [http://www.w3schools.com/default.asp](http://www.w3schools.com/default.asp). Along with character references, we can include standard HTML code in our SAS code such as `<sup>XXX</sup>` for superscripts and `<sub>XXX</sub>` for subscripts. The following invocation produces the output shown in Figure 1 for using special character.
NUMBER (%) OF SUBJECTS WHO DISCONTINUED STUDY BY PRIMARY REASON

<table>
<thead>
<tr>
<th>Conclusion Status</th>
<th>Overall P-Value</th>
<th>Treatment A</th>
<th>Treatment B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discontinued</td>
<td>&lt;0.001***</td>
<td>7 (70.0)</td>
<td>3 (33.3)</td>
</tr>
<tr>
<td>Adverse Event</td>
<td>0.12</td>
<td>1 (10.0)</td>
<td>3 (33.3)</td>
</tr>
<tr>
<td>Discontinuation of Study by Sponsor</td>
<td>1.000</td>
<td>1 (10.0)</td>
<td>1 (11.1)</td>
</tr>
<tr>
<td>Lost to Follow-up</td>
<td>0.039*</td>
<td>8 (8.0)</td>
<td>1 (11.1)</td>
</tr>
<tr>
<td>Protocol Violation</td>
<td>0.839</td>
<td>2 (14.3)</td>
<td>1 (11.1)</td>
</tr>
<tr>
<td>Subject Request</td>
<td>0.37</td>
<td>1 (10.0)</td>
<td>1 (11.1)</td>
</tr>
</tbody>
</table>

a. Total discontinued is the sum of individual reasons since they are mutually exclusive by subject.
Overall P-Value: Fisher's Exact Test P-value (2-Tail).
Statistical significance at the .05, .01, .001 levels is denoted by *, **, *** respectively.

Figure 1. SUMMARY OF SUBJECTS WHO DISCONTINUED STUDY BY PRIMARY REASON

Example 2: ASCII Formatting and By Statement

Any ASCII formatting (such as using spaces to align characters in a column) will not work using the HTML tagset. This is because HTML’s behavior is to collapse all consecutive whitespace to a single space character. All alignment should be done through the standard mechanisms of the output producing procedure, which will in turn pass the information to the tagset. In HTML, &nbsp; is the non-breaking space character to be used in place of a normal space, when multiple spaces should not be collapsed. The HTML line break character "<BR>" is inserted into data to get line break instead of using “|”. When using a by variable, we should specify “options nobyline” which is required by our standards. This invocation produces the output listing shown in Figure 2, listing of SERIOUS ADVERSE EVENT, which will be inserted into the clinical study report.

%getmacro (ldtagset);
data _report_
   set _report_
   eventcls1 = tranwrd(eventcls,'|','<br> &nbsp&nbsp;' )
run;

options nobyline pageno=1 missing=" ";
ods markup tagset=html.standard body="&htmlfil";
proc report nowd data=_report_ split='|' nocenter headline missing spacing=1;
   by invtext tpname;
   column patient agesex tese fsaelyn1 eventcls1 bs rel sev action outc abs_rel strtdate stopdate duration dai;

CPP5_D - 27NOV06 09:50
define patient / left width=7 order "&subpat|NUMBER" ;
define agesex / left width=4 order "AGE/SEX" ;
define tese / width=1 "T|E|A|E" ;
define fsaeyn1 / width=1 "S|A|E" ;
define eventcls1 / left width=&_w_ flow "ADVERSE EVENT|VERBATIM";
define bs / left width=4 "BDY|SYS" ;

Skip other part of the program

ods markup close;

REPORT AE1_SAE LISTING OF SERIOUS ADVERSE EVENTS BY INVESTIGATOR AND TREATMENT GROUP

| INVESTIGATOR: XXXXX |
| TREATMENT : A |

<table>
<thead>
<tr>
<th>SUBJECT NUMBER</th>
<th>AGE</th>
<th>ADVERSE EVENT</th>
<th>VERBATIM</th>
<th>BDY</th>
<th>RELATION</th>
<th>SEVERITY</th>
<th>ACTION</th>
<th>OUTCOME</th>
<th>STUDY DAY</th>
<th>ONSET DATE</th>
<th>STOP DATE</th>
<th>DURATION</th>
<th>ONSET INTERVAL</th>
</tr>
</thead>
</table>

NOTE: STUDY DAY IS THE ELAPSED DAY OF THE ONSET DATE RELATIVE TO FIRST DAY OF STUDY MEDICATION.

AE1_SAE - 28NOV06 15:18

Figure 2. LISTING OF SERIOUS ADVERSE EVENTS

Example 3: Aligning data using HTML tagset

The CHTML tagset does not adhere to any style information passed through PROC TEMPLATE. The following code will not work if we use the ldtagset.

```
define name / order style(header)=[justify=left]
    style(column)=[cellwidth=1in];

ods html close;
```

For certain portions the alignment is overridden in our tagset. If we need to change the alignment, we should wrap the value with the text:

```
<div style="text-align: XXX"> value </div>
```

where XXX is one of the following: left, right, center, justify and value is the value that is to be aligned. The following code produces the output listing shown in Figure3, NUMBER (%) OF SUBJECTS EXCLUDED FROM VALID FOR ANALYSIS POPULATION, which will be inserted into clinical study report. This example code aligns the header of variable sasname by wrapping it with the HTML tag pair "<div style='text-align:left'>" and "</div>"

```
option nobyline;
ods markup tagset=html.standard body ="../output/&htmlfil";
```
REPORT VFA4_BLDPP NUMBER (%) OF SUBJECTS EXCLUDED FROM VALID FOR ANALYSIS POPULATION
FOR BLEEDING DATA BY PILL PACK

<table>
<thead>
<tr>
<th>DATA ANALYSIS INTERVAL</th>
<th>CRITERIA DESCRIPTION</th>
<th>Treatment A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pill Pack 14</td>
<td>VALID DATA ANALYSIS INTERVAL</td>
<td>77/78 (99)</td>
</tr>
<tr>
<td></td>
<td>INVALID DATA ANALYSIS INTERVAL</td>
<td>1/78 (1)</td>
</tr>
<tr>
<td></td>
<td>&gt;=1 DAY BLEEDING DATA MISSING</td>
<td>1/78 (1)</td>
</tr>
<tr>
<td>Pill Pack 15</td>
<td>VALID DATA ANALYSIS INTERVAL</td>
<td>77/78 (99)</td>
</tr>
<tr>
<td></td>
<td>INVALID DATA ANALYSIS INTERVAL</td>
<td>1/78 (1)</td>
</tr>
<tr>
<td></td>
<td>&gt;=1 DAY BLEEDING DATA MISSING</td>
<td>1/78 (1)</td>
</tr>
</tbody>
</table>

N REPRESENTS THE NUMBER OF SUBJECTS IN EACH THERAPY GROUP INCLUDED OR EXCLUDED AND NUMBER EXCLUDED FOR EACH CRITERIA.
A SUBJECT MAY HAVE BEEN EXCLUDED FOR MORE THAN ONE REASON.
VFA4_BLDPP - 26JUN06 10:23

Figure 3. NUMBER (%) OF SUBJECTS EXCLUDED FROM VALID FOR ANALYSIS POPULATION

CONCLUSION

At this point you should have an idea of how easy it really can be to change the markup output generated by ODS tagset. You may already be thinking about what you want to take advantage of ODS advanced features to produce a standard template to control the color, font, size, justification, order and labels, the format of the output report with the guides from the Submission Ready Components Standard for in-text tables. You should never have to write a complete tagset. By re-writing several events and, possibly, adding options and functions, significant changes in output can be accomplishes. Using ODS to generate HTML output is an effective method of incorporating SAS output in Excel and Word documents.
REFERENCES


TRADEMARKS

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.

ACKNOWLEDGMENTS

The author would like to thank Mr. Michael Jessup at Wyeth who developed the template ldtagset and updated the template of feedback from users.

CONTACT INFORMATION

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

Li Luo
Wyeth Pharmaceuticals
500 Arcola Road
Collegeville, PA 19426
(484) 865-5896
(484) 865-4322(Fax)
luol2@wyeth.com