An Application of ODS Tagsets

Paper 178-27

Jack Hamilton
First Health

SUGI 27
SAS® Users Group International
Orlando, Florida • April 14–17, 2002

Notice!

- For detailed code, look at my paper in the Proceedings.
- I have taken to heart Edward Tufte's advice that there's no point in just reproducing the paper in the oral presentation. Read the paper and the SAS documentation for details.

Overview

- I want to give you an overview of tagsets. I'm not going to go into a lot of detail. See the proceedings if that's what you want. It will be easier to absorb if you have it in writing.
- Don't worry if this doesn't make sense the first time you hear it. The way the parts work together makes sense, but it might not be obvious at first.

ODS

- The Output Delivery System is the SAS System's way of allowing you to tailor your output. You can specify a destination, a style, and how individual elements will be displayed.

ODS Destinations

- ODS destinations describe how the output will be displayed:

```
ods listing;

ods pdf file=_webout;

ods html
     file='c:\temp\my.html';
```

- Each destination statement creates an Output Delivery Agent.

ODS Styles

- ODS Styles describe in more detail what the output should look like when displayed:

```
ods html file='~hamiltja/test.html'
   style=printer;
```

What's Wrong With “Out Of The Box”?

- In my application, which is described in more detail in the Proceedings, I wanted more control over output than I could get with the standard styles. In particular, I wanted:
  - Very plain HTML.
  - An easy way to add lots of <head> material to HTML.
  - An rich-content plain text format.
- SAS doesn't provide any of those.

Why Not Use Templates?

- PROC TEMPLATE provides the capability of modifying existing ODS descriptions or creating new ones. It's well and extensively documented by SAS in *The Complete Guide to the SAS Output Delivery System*, and in Lauren Haworth's *Output Delivery System: The Basics*. I could have created the HTML part of my application without using tagsets.

Because They're Too Hard!

- But there's a fundamental problem, for me, with templates: *I don't understand them well enough to feel comfortable using them*. I can modify bits here and there to get what I want, but I don't have "the big picture". It's too complicated.
- I have a much better feel for tagsets, even though they're much newer, and that's why I chose them – they're like programming.
- YMMV

So What's A Tagset?

- A tagset is a way of describing how to create output from ODS. It's basically an event-driven programming language.
- Event-driven means that a piece of tagset code is executed, or triggered, when some relevant event happens in ODS, rather than starting at the top and going through the code to the bottom, as in a traditional SAS program.

Event Driven Programming?

- It sounds like a big deal, but it's not, really. The leap between the data step (procedural) and SQL (declarative) is much big than the leap between procedural and event-driven.
- If you've used VBA in Microsoft Word or Excel, you've already done event-driven programming – and the event models there are much more complicated.

An Event And Its Code

- An example of an event which occurs in HTML output is *top_file*. It occurs, naturally enough, when the top (first) part of your output file is being written.

```
define event top_file;
  put '<html>';
end;
```

What Events Are Available?

- Events are available for anything that results in output. I’m not sure of the exact number, but there are dozens.
- But don’t worry – you won’t use most of them. My tagset for plain XHTML from PROC REPORT uses only 10 events, and I could have used fewer than that.

What Events Do I Need?

- To get a list of the events which occur during the creation of your output, use the event_map tagset:

```
ods markup type=event_map
    body='c:\temp\event.xml';
```

- It creates a text file which you can review to see which events occurred.
- Other types are short_map & text_map.

Another Way To Proceed

If there's an existing tagset which is almost what you want, just get the source code and modify it.

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

Gives me something that's pretty close to my XHTML code.

What Commands Are Available?

The tagset language allows you to:

- Write out variables or constant text using the PUT, PUTL, and PUTQ statements.
- Conditionally execute statements using the IF condition and its relatives.
- Indent output using NDENT and XDENT.
- Call subroutines using TRIGGER.

One of the design goals was simplicity, so the syntax is not complicated.

What Information Is Available?

The tagset code has access to data:

- Many events have a value associated with them. For example, when a cell is being written out with the DATA event, the variable VALUE contains the value in that cell.
- Some events create additional variables. You'll get a list when you run an event map.
- Some values, such as the HTML title, are available generally.

Tagsets in Action

```
define event doc_title;
   put  '<title>';
   put  VALUE;
   put  '</title>' nl;
end;

define event doc_head;
   start:
      put  '<head>' nl;
      ndent;
   finish:
      xdent;
      put  '</head>' nl;
end;

define event doc_body;
   start:
      put  '<body>' nl;
   finish:
      put  '</body>' nl;
end;
```

```
<head>
   <title></title>
</head>
<body>
```

Tagsets in Action

```
define event table;
   start:
      put
 '<table border="1">' nl;
      ndent;
   finish:
      xdent;
      put  '</table>' nl;
end;

define event row;
   start:
      put  '<tr>' nl;
      ndent;
   finish:
      xdent;
      put  '</tr>' nl;
end;
```

```
<table border="1">
   <thead>
      <tr>
         <th>ZIP3</th>
         <th>c_iwc</th>
         <th>c_gs</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>403</td>
         <td>        123</td>
         <td>        456</td>
      </tr>
```

Tagsets in Action

- The only thing that's complicated in this tagset is the code which decides whether to write <th> or <td>.

```
define event data;
   start:
      put  "<th>"
        / if  cmp(section, "head");
      put  "<td>"
        / if !cmp(section, "head");
      put  VALUE;
   finish:
      put  "</th>" nl
        / if  cmp(section, "head");
      put  "</td>" nl
        / if !cmp(section, "head");
end;
```

A Reminder…

- Tagsets are event-driven.
- Which events are executed and in what order is controlled by the procedure creating output, not by your code.
- So… if you want to rearrange the order of your output, you'll have to take some additional steps.

Conclusion

- Tagsets add flexibility to the output styles which ship with SAS.
- Tagsets are a bit intimidating at first, but not all that difficult to write.

Acknowledgements

- Sandy McNeil, SAS Institute
- Brian Schellenberger, SAS Institute
- Eric Gebhart, SAS Institute

Questions?

About the Speaker

Speaker	Jack Hamilton Development Manager Metrics Department
Company	First Health 750 Riverpoint Drive West Sacramento California 95605 USA
Telephone	(916) 374-3833
Email	jackhamilton@firsthealth.com