

## Using Embedded HTML in SAS/Intrnet® Output

Bill Dvorak  
Northwestern Mutual Life

**Abstract:** SAS/Intrnet® developers can improve the navigation and the usability of their web sites by embedding HTML in the output created by HTML formatting macros. These improvements aid the transition of applications from static printed reports to dynamic systems. This paper will discuss using PROC TABULATE to create a table of contents page that serves as an HTML form to call another SAS® program. Using this technique, the developer can create drill-down capabilities for users.

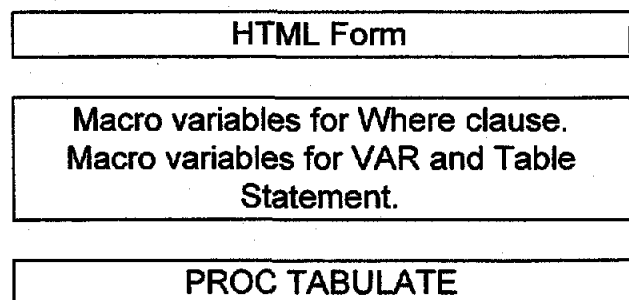
### Background

MIS reporting is a major portion of our area's work. Historically these reports have been printed and manually delivered. Considerable time and energy went to generate and deliver these reports on daily and weekly basis. Over time, as management issues changed, new reports were created while the existing reports continued to be generated. The inertia to respond to new issues overwhelmed the desire to weed out existing reports whose time had passed. The development of a corporate Intranet and the availability of SAS/Intrnet made it possible to develop and distribute interactive systems. This paper deals with the transition of a static reporting system to a dynamic report generating application. I'll discuss the criteria that made the application appropriate for SAS/Intrnet. And lastly, I will present some techniques that

helped deliver the information in a form that is compatible with a browser based environment.

### Going from Static Reports to an Interactive System

Not all reports are suitable candidates to be moved to an interactive system. We *did not* want 30 managers all generating an identical report each morning. In this situation, it would be better to generate the report once and publish it as a static HTML page. The interactive nature of SAS/Intrnet comes into play when the report that is generated is customizable. The approach that we took was to create SAS programs that were boiler-plate templates. An HTML form created macro variables to be used by the SAS program as the selection criteria and the variables that would be displayed. The flow of this system looks like this:



As we analyzed how the managers used the reports, we found they used the summary reports to find problem areas, either with an individual or a unit. Then they would use the detailed

listings to pinpoint specific cases that needed more attention. This meant that the managers would use a small portion of the hundreds of pages of reports that they received. On the summary reports they concentrated on conditions that they were concerned with and then only using the detail for those cases that met those conditions. For example, they might be interested in cases that were more than 10 days old. A summary report would show the number of cases per underwriter broken out into different duration categories. They would then page through a detailed listing of all cases sorted by underwriter for those underwriters who appeared to have a larger than normal number of cases that were over 10 days old. The majority of the information that we provided was not used, they received everything only because they could not anticipate what conditions they might be interested in. The SAS/Intnet application would allow them to choose what conditions they wanted to monitor and also provide details on cases that met those conditions. By culling out the unwanted information we were able to reduce the amount of output so that it would be suitable to be viewed with a browser. Very few of these reports now generate more than a hundred lines of output for any particular user.

### **Phase I – The SAS Boilerplate Program**

The first design for the boiler-plate SAS application was a concatenation of PROC TABULATE output followed by DS2HTM output (the HTML equivalent of PROC PRINT). The manager would select the desired conditions from an HTML screen and submit the form to

SAS/Intnet. SAS would generate a cross-tabulation by underwriter and a detail listing by underwriter using the WHERE clause and variables that were passed as macro variables. The manager could look at the table at the top of the report and then scroll down to see the detail for any of the underwriters listed. The schematic of the SAS program is:

#### **DATA STEP**

- Subsets data.
- Does calculations.
- Sets up macro variables for titles and footnotes.

#### **PROC TABULATE**

- Output is captured by %TAB2HTM formatting macro.

#### **%DS2HTM**

- HTML equivalent of PROC PRINT

### **Phase II – Using Internal HREFS for Navigation**

The phase I program provide useful output but scrolling through the output was cumbersome as was using the Find feature in the browser. I decided to add internal hyperlinks within the document by embedding the HTML tags necessary to create HREF links in the PROC TABULATE output that would correspond to anchors in the DS2HTM detail listing. This was fairly easy to do by creating new character variables in the DATA step, one variable appended the necessary HTML to create anchors for the listing and another variable had the HTML to create the links. The

PROC TABULATE would use the link variable as the class variable and the DS2HTM would use the anchor variable as a BY variable. The manager could now click on the underwriter name in the table and jump directly to the detail listing for that underwriter. This method uses the navigation that is inherent in the browser and makes it less likely that the manager will need to print the output to reference different sections.

HTML allows for internal hyperlinks within a page. These links allow the user to jump between sections in a long document. The HTML for creating these links uses this form:

To create the anchor point

```
<A NAME="var">text</A>
```

To create the hyper-link

```
<A HREF="#var">text</A>
```

The SAS code needed to implement these hyperlinks is:

To create the anchor for each occurrence of the variable undrwtr:

```
und_tag = '<A NAME="' || trim(undrwtr) || '"'>' ||  
undrwtr || '</A>';
```

The variable und\_tag will be used in the DS2HTM macro as a BY variable. This creates a section for each underwriter that has the anchor at the top of each section.

To create the link for the variable undrwtr:

```
und_link = '<A HREF="#' ||  
trim(undrwtr) || '"'>' || undrwtr || ' </A>';
```

The variable und\_link is used in the PROC TABULATE as a CLASS variable. I use it as the row class.

In the DS2HTM and TAB2HTM formatting macros, use the option ENCODE = N. This will pass the embedded HTML in the und\_tag and und\_link variables in the HTML page. The browser will then use the internal anchors to create the hyperlinks between sections.

### **Phase III – Using the Output of PROC TABULATE to Generate an HTML Form**

The final incarnation of this program breaks the PROC TABULATE and DS2HTM into two separate programs. The PROC TABULATE output would serve as an HTML form and is used to call the DS2HTM program for detail listing. There are several advantages to this technique:

- The output of the PROC TABULATE is generate faster.
- The DS2HTM generates only the detail for the desired underwriter.
- The DS2HTM output is shorter and prints better from the browser.

The schematic of the phase III technique is

#### **DATA STEP**

- Subsets data.
- Does calculations.
- Sets up variables for HTML radio buttons.

#### **TITLE and FOOTNOTE statements**

- Used to generate HTML for HTML form to call SAS program.

### PROC TABULATE

- Output is captured by %TAB2HTM formatting macro.
- Has embedded HTML for radio buttons and HTML form.

### Intermediate HTML Page from PROC TABULATE

- Calls DS2HTM SAS program.

### %DS2HTM

- HTML equivalent of PROC PRINT

The SAS code to generate the HTML radio button based on variable undrwtr is:

```
und_link = '<INPUT TYPE="RADIO"
NAME="level" VALUE="" || trim(undrwtr) || ">' ||
undrwtr;
```

The variable und\_link is used as the CLASS variable in the PROC TABULATE. When the button is clicked the value for undrwtr is passed in the macro variable &level when HTML form is submitted to call the DS2HTM program. The &level macro variable is used in the DS2HTM program as a condition in a WHERE clause.

Title and Footnote statements are used to pass the HTML that is necessary to create an HTML form and also have the HTML form have the required values to call a SAS/Intrnet program.

Title2, Title3 and Title4 have the HTML that is used to call a SAS/Intrnet program called UNDWINTR.SAS.

```
Title2 '<FORM ACTION="http://webapps1/cgi-
bin/broker.exe" METHOD="GET"> <INPUT
```

```
TYPE="HIDDEN" NAME="_program"
VALUE="newbusns.undwintr.sas">;
```

```
Title3 '<INPUT TYPE="HIDDEN"
NAME="_service" VALUE="default">;
```

Title4, Title5 and Title6 carry the original macro variables forward so that the same selection criteria used to generate the PROC TABULATE are used to create the detail listing.

```
Title4 "<INPUT TYPE='HIDDEN' NAME='durvar'
VALUE='&durvar'>;
```

```
/* pass date selection */
```

```
Title5 "<INPUT TYPE='HIDDEN' NAME='low'
VALUE='&low'>;
```

```
/* pass low end for duration */
```

```
Title6 "<INPUT TYPE='HIDDEN' NAME='high'
VALUE='&high'>;
```

```
/* pass high end for duration */
```

Title7 and the Footnote statement create the Submit button and some other buttons for the HTML form. The footnote ends the <FORM> tag, this caused some problems in Internet Explorer 3.

```
Title7 '<INPUT TYPE="SUBMIT" NAME="TEST"
VALUE="Get Detail"> <INPUT
TYPE="RADIO" NAME="reqsvar" VALUE="N"
CHECKED> Show All <INPUT
TYPE="RADIO" NAME="reqsvar" VALUE="Y">
Only Reqs. Met;
```

```
Footnote '</FORM>;
```

Once again, the option ENCODE = N is used in the TAB2HTM and DS2HTM macros so that the HTML is passed to the browser. It should also be noted that the RTS option in the Table statement can cause problems with this technique. If it is set too short, the variable value will wrap and TAB2HTM

will put a <BR> tag at the wrap point. This will interfere with any embedded HTML.

The phase III version has proven to be very popular because it gives the managers the ability to "drill-down" to show a detailed listing without generating all the detail for all underwriters. However, the phase II version remains in use because it generates all the summary and detail in one pass and allows the manager to jump around much faster. If the manager wants to look at several underwriters at a time, the phase II program is used.

### **Using Embedded HTML to Highlight Data**

One last technique that I want to pass along is using embedded HTML to highlight output in the HTML pages. In our application, we wanted to highlight any case where all requirements had been met. We had a flag variable that is set if all requirements are met. In the detailed listing, we wanted the field to show in red if this flag was set. We used the following format to embed the HTML in the DS2HTM output.

```
Proc Format;
value $reqhtml
'0' = 'No'
'1' = '<FONT COLOR="RED">Yes</FONT>'
```

The variable reqsmet was then formatted using \$reqhtml. This embedded the <FONT> tag in the output so that all "Yes" displayed in red.

### **Summary**

So far, the SAS/Intrnet product has proven to be very valuable. By moving parts of our daily and weekly reports to a SAS/Intrnet application we have reduced the amount of printed output by more than 30%. Our managers are also very enthusiastic about using this application because it has reduced the amount of information they need to filter out to get to the information they need. One key component in this application has been the navigation capabilities we've created by embedding HTML in the output. Use of hyperlinks and dynamic forms reduces the amount of output created by any program to manageable chunks that can be viewed with the browser and avoid the need to create hard copy.

### **Acknowledgements**

SAS and SAS/Intrnet are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

I wish to acknowledge the assistance of Steve Mageski and Scott Seifert in their help in installing SAS/Intrnet on our company intranet.

Bill Dvorak  
Northwestern Mutual Life  
720 E. Wisconsin  
Milwaukee, WI 53202

e-mail: bdvorak@yahoo.com