Update HTML Formatted SAS Output Within Web Browser

James Sun, Constat Systems, Monmouth Junction, New Jersey

ABSTRACT

In order to get most recent information, SAS programmers are frequently asked by end users to rerun some existed programs. This paper presents a step-by-step method to setup a light-weight SAS and intranet application, allowing end users to update SAS output within their browser. The author highlights the mechanism of such applications. Techniques discussed include setting up personal web server, selecting proper CGI script, and modifying existed SAS program in order to successfully deploy the application. Three examples covered illustrate different ways to use such an approach.

INTRODUCTION

SAS outputs are often presented in hard copy. To view outputs on screen, such as in AF or EIS application, requires SAS running. It is natural to ask this question: Can we take advantage of versatile HTML format to deliver the information SAS generated in a timely fashion?

With some basic knowledge of HTML format, we can translate a regular SAS output into HTML document. But a converted SAS output is just a static document. It only reflects the data when you submitted the SAS job. When asking for latest report, SAS program should be rerun with updated data. It is inefficient to have programmers to handle all update requests. To relief the burden on SAS programmers, end users should be able to rerun the SAS code themselves. In some situations, those end users are not in a SAS installed environment.

On the other hand, it is complicate to develop a traditional client-server application without install SAS at client site. To overcome this problem, we can utilize the network facility to set up a light-weight intranet. An ‘update’ button could be placed within SAS output file. Whenever end users want to access updated information, all they need to do is click the “Update’ button. A personal web server resided at programmer’s desktop will invoke a batch job running the SAS program.

Three examples covered in this paper are mostly experimental. They were designed to assist in monitoring the progress of data processing. The levels of difficulty are ranging from simple, only requiring minimal setup and changes, to sophisticate which produced integrated report. The last example goes beyond the simple ‘update’ function. It is actually a multilevel SAS intranet application.

HTML FILE AND SAS OUTPUT

A HTML file looks like contents of document, plus some HTML tags that indicate document elements, structure, formatting, and hypertext links.

A straight forward method to put SAS output into a web browser is using PUT statements in DATA steps. The output includes all necessary HTML tags mixed with main contents.

The macros %OUT2HTM, %DS2HTM and %TAB2HTM introduced by SAS Institute (SI) put all necessary HTML tags into the output. Those handy tools simplify the process of delivery SAS output through intranet.

SAS WORKS WITH WEB

Two parts are needed to run a SAS program in an intranet environment. At end user side, the only requirement is a web browser. The purpose of using a browser is twofold. It acts as a SAS output reader, as well as a user front end. It let user to issue rerun command.

The second part is installed in the programmer’s desktop. It includes three components:

- Web server. It should be able to accept CGI script. Because the application illustrated here assumed works in an environment which no formal intranet was previously installed. Most of personal web servers meet this requirement.
- CGI script. Common Gateway Interface is a piece of executable program which reside inside

web server. It response to the request received by server to executive certain system tasks. In this application, the CGI will be responsible for starting SAS and then running a designated program.

- SAS system and program. The center piece of this application is the SAS program which creates the output. Adapting to this setting, the original program should be modified accordingly.

The diagram below is the setup and data flow.

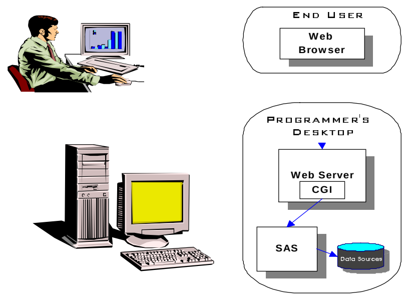


Figure 1 Setup and Data Flow

First of all, SAS programmers need to generate the HTML formatted report. This output is physically stored in a shared network storage, which can be access by end users. It can be viewed with a browser. When comes to update, the end users just need to click the button embedded in the report. After the request reach the web server, a batch SAS session then executes the SAS program. End users will receive the update output in browser as the rerun completed.

SET UP APPLICATION

The requirement at site of end user is minimum. Only web browser is need. Most of setup work will be at site of SAS programmers, who are previously responsible for carrying out the update tasks.

Install Web Server

Installing personal web server is straight forward. You only need to provide information about your network, such as your IP address.

There are many commercial web software as well as shareware available for different operation systems. Some of them allow you to download directly from Internet .

Select CGI script

For the purpose of illustrating how it looks like, a very basic example of CGI written in a Windows batch file is following.

```
@echo off
 echo HTTP/1.0 200 OK>>  %3
 echo Content-type: text/html>>  %3
 echo.>>  %3
 type %2 > c:\test\stdin
 c:
 cd c:\sas612
 start /w sas.exe -sysin c:\test\update1.sas -nologo  -icon
 type c:\test\jump.htm >>  %3
```

CODE 1 Example of a Simple CGI

The CGI example above is too primitive to use in practical situation. Some CGI scripts have many enhanced functions. They are capable of better communication between server and SAS system. If you choose CGI script written in Perl, an interpreter for Perl should be installed first.

Modify SAS program

For simple tabulate report, such as printout of dataset, you can choose macros included in SI’s Web Publish Tools package. For the program using SAS/GRAPH procedures, GIF or JPEG device driver is required. In order to create composite (text mix with graphic) output, HTML knowledge is essential.

The embedded clickable ‘Update’ button is accomplished with HTML‘s FORM. You can place the button anywhere you want. It is much simple to use TITLE or FOOTNOTE statement to place the button on top or bottom respectively. It is important to put the date indicating when the output generated. If prefer better integration of button and output, you can use PUT to insert those HTML code. The next SAS code example would place the button beneath the output titles (TITLE1-TITLE4).

```
TITLE5 ‘<CENTER><FORM ACTION=”http://129.129.
129.129/Cgi-bin /runsas.pl” METHOD=”post”>’;
 TITLE6 ‘<INPUT TYPE=”hidden” NAME=”pgm”
VALUE=”sugi23.sas”>’;
 TITLE7 ‘<INPUT TYPE=”SUBMIT” VALUE=”Update Now”
></FORM></CENTER>’;
```

CODE 2 SAS Code for a "Update" Button

Within the code, ‘129.129.129.129’ is the IP address of programmer’s desktop. ‘Cgi-bin’ is the name of directory CGI stayed. ‘runsas.pl’ is the CGI script I used. Obviously, you need to adjust them base on your IP address and server setup. Some of CGI scripts will allow you to use extra input to specify the program name. It avoids clogging the ‘Cgi-bin’ subdirectory with all sorts of different CGI scripts in case of running different SAS programs. TITLE6 provides the name of program.

The HTML formatted output will be stored in a shared network driver. Hence, end users can access the SAS results any time they want. After an update completed, a jump page will then automatically load the new output after a short pause.

```
DATA _NULL_;
   FILE PRINT NOTITLE;
   PUT 'Content-type: text/HTML';
   PUT ;
   PUT '<HTML>’;
   PUT ‘<META HTTP-EQUIV=”Refresh” CONTENT=
”5;URL=file:///G|\sasout\output1.html” ’;
   PUT '<BODY ><H1> Update Finished! </H1>’;
   PUT '</BODY></HTML>';
RUN;
```

CODE 3. SAS Code for a "Jump" Page

CODE3 creates a jump page. It will keep this page for 5 seconds before display the new output, assuming G is the shared driver, and output1.html is the updated output.

If rewrite CODE2 and CODE3 into SAS macros, “output1.html” and “sugi23.sas” can be represented by macro parameters.

EXAMPLES

Ex1. *Simple Data Output*

This output is a progress report of a tracking application. In clinical trial, data are collected on each patient’s clinical report form (CRF). During an on-going trial, CRFs filled at each visit are sent to a data center for processing. The SAS report presents date when CRF received and other status.

The data source is from a SAS/FSP application running at different group. It is accessed through SAS/SHARE allowing concurrent usage.

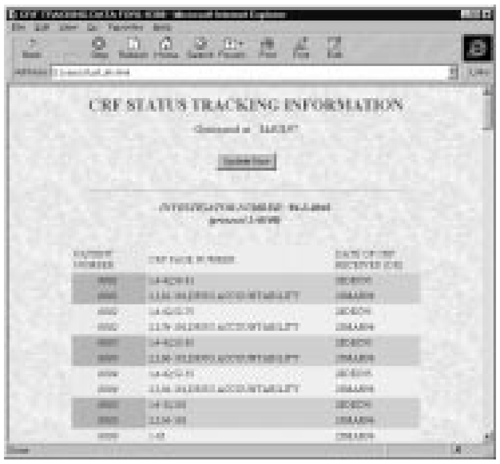


Figure 1 CRF Tracking Information

To schedule a rerun in advance is impractical. Since CRF status tracking data was updated irregularly. It is an example of using this approach to enable end users control the timing of update.

```
/*-----  prepare dataset ‘OUT’  ------*/
              .......
/*-----  generate HTML output  ------*/
OPTIONS PS=1000;

TITLE1 ‘<HTML><HEAD><TITLE>CRF TRACKING
</TITLE></HEAD><BODY BACKGROUND=back1.jpg>';
 TITLE2 '<CENTER><H1><FONT COLOR=red> CRF STATUS
TRACKING INFORMATION </FONT></H1> </CENTER>';
 TITLE3  “<CENTER><H3><FONT COLOR=RED> Generated at
:  &sysdate</FONT></H3></CENTER>”;

%UPDATE(sugi23);

FILENAME NEWOUT ‘G:\SASOUT\CRF_TRK.HTML’;


%HTMOUT(DATA=OUT,
                BY=INV,
               VAR=PAT PGNO DATENT,
               COLWIDTH=20 65 25
               LABCOL=EEDDEE EEEEAA EEAABB
               ROWCOL1=99CC99 CCCCCC CCCCCC
              ROWCOL2=CCFFCC EEEEEE  );

/*----- create jump  page   ---*/

FILENAME webout  "&outfl";

%JUMP(OUT=webout,
URL=%NRSTR(file:///G|\sasout\crf_trk.html), );
```

CODE 4 SAS code for Case 1

The purpose of large page size in OPTIONS statement is to make sure that only one appearance of title section within the whole output. The first 3 titles include HTML header information and output title.

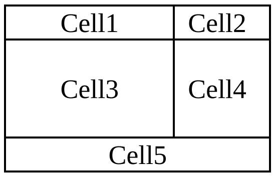
The macros %UPDATE and %JUMP contain the code shown in CODE2 and CODE3. The macro named as %HTMOUT, similar to SI's %DS2HTM, translates SAS dataset into HTML table. "&outfl" is required by the CGI script. The figure 1 is the result.

Ex2. *Data and Graphic Composite Output*

The next example is an output containing not only the main graphic plot, but some precise data results as well.

To display graphical result in a web browser, SAS/GRAPH procedure should run first. The converted GIF or JPEG file is saved in network storage waiting to be called upon.

To produce this composite report, the SAS program first creates the plot named 'graph1.gif'. Then PUT statements are used to generate the output file based on the layout design. Figure 2 shown the design and basic HTML file. This file will be the major structure generated by PUT statements.



```
<TABLE>
<TR><TD>Cell1</TD><TD>Cell2</TD></TR>
<TR><TD>Cell2</TD><TD>Cell3</TD></TR>
<TR><TD COLSPAN="2">Cell5</TD></TR>
</TABLE>
```

Figure 2 Basic Layout and HTML Code

In this example, Cell3 was designed for the plot, so tags of '*<IMG SRC=" file:///G|\sasout \graph1.gif">*' replaces cell3 in the output. With the update button is placed inside Cell5, instead of using TITLE or FOOTNOTE, the PUT statements will lay those HTML code (for the button) into the place occupied by cell5.

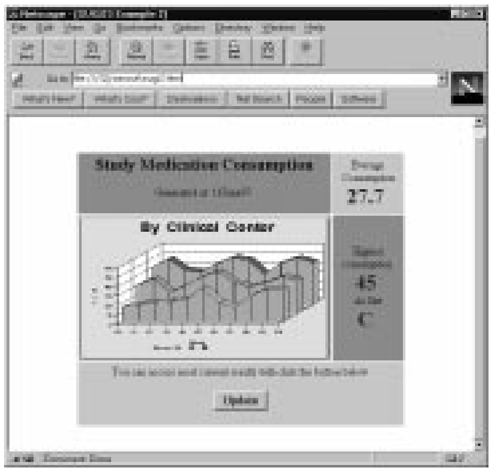


Figure 3 Medication Consumption

To deal with dynamic contents inside other cells, macro variables are employed. For an example, the dynamic content - current average consumption number in Cell 2 will be represented by a designated macro variable. This macro variable is generated with CALL SYMPUT in an appropriate data step. When the PUT statement try to put the macro varible into the output file, it will automatically render its resolved value.

After first two parts of SAS program, one generates graphic another produces the HTML output, the last part of the program is for a jump page. A filename statement required by the specific CGI script and macro %jump conclude the whole SAS program.

One benefit of preparing graphics for web browser, is able to avoid tedious ANNOTATE or DSGI coding in creating composite output. Another advantage is the capability of creating animation graphic with special SAS device driver of GIFANIM.

Ex3. *Patient Count Information*

Compare to the previous examples, this one is more complex. Not only allows end users control the timing of rerun, it also provides ways to further investigate the completeness of data.

In clinical trial, it is a common practice to separate patient data into different data sets. Often some patient data are missing in certain datasets. The purpose of this example is to detect those missing cases based upon the discrepancy among the datasets.

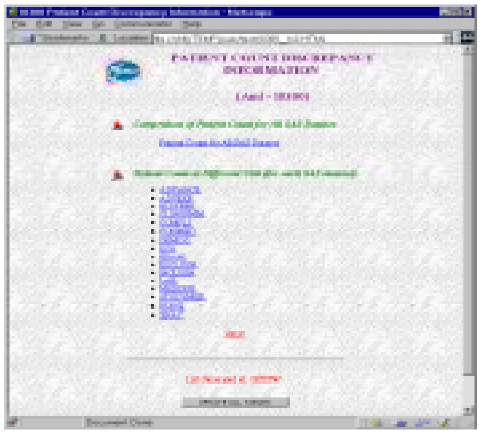

PATIENT COUNT DISCREPANCY INFORMATION

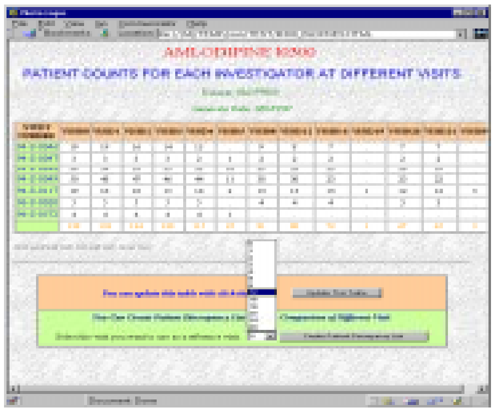
Figure 4 List of Available Tables

The application was designed to work for all studies which follow a common data structure. At first, end users are asked to setup the application by filling a form with their browsers. The application then generates a set of tables to check following:

1. Patient's existence among all SAS datasets
2. Patient's existence at all visits within each SAS dataset

Figure 4 is the master list. It indicates all available tables. An update button is placed on the list. End users can rerun all the tables when they feel necessary.

Figure 5 is one of those tables. It provides patient counts in all dataset or patient counts at different visits for a particular dataset. The counts are sorted by investigator sites. Beneath the table, a drill-down list generated by SAS code is essential to further detect which patient is actually missing.

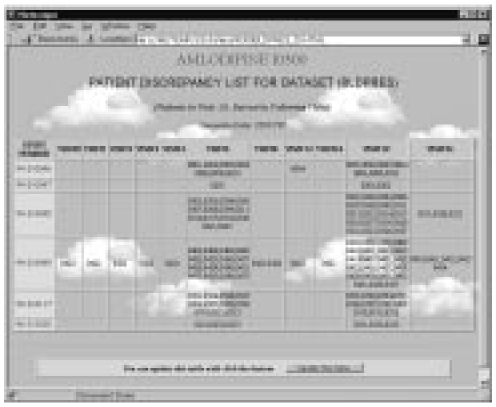

AMLODIPINE 10500

PATIENT COUNTS FOR EACH INVESTIGATOR AT DIFFERENT VISITS

Figure 5 Patient Count in One Dataset

The drill-down list was constructed with <SELECT>, <OPTION> and realeted HTML tags. All the available choices inside the drill down list are created by SAS code based upon the column values. By the side of drill down list there is another submit button. It will invoke a different SAS program which produce the discrepancy list.

Once user select an anchor dataset or visit with the drill-down list, the application can produce a listing of patient IDs. Those IDs indicate who are missing comparing to the anchor dataset or visit. Figure 6.is the listing which provides patient missing information at different visits when compare to the situation at Visit20.


AMLODIPINE 10500

PATIENT DISCREPANCY LIST FOR DATASET (R.DPRES)

Figure 6 Discrepancy Patient List

Other than the chosen anchor dataset name or visit number, the batch SAS program still need to know project parameters and even dataset user wants to further investigate. To overcome shortfall in this typical 'stateless processing', additional macro variables are provided with <INPUT TYPE="hidden" … > HTML code. Therefore, SAS programs are able to execute accordingly.

DISCUSSION

The technique illustrated here leverages TCP/IP network to facilitate an alternative information deliver method. The network environment grants the possibility of extending the reach of SAS outputs. By enable end users to update SAS result, the application lessen the needs of SAS programmer in maintaining programs. Also such applications can be easily integrated into department-wide intranet.

Because of the experiment nature of these examples, there are limitations in this approach. The use of personal web server is only suitable for small scale deployment. Heavy use of applications such as OLAP or Data Warehouse application demands commercial grade server together with SAS/IntrNet.

SAS/IntNet uses different working model comparing to that of CGI script. Instead of launching SAS every time, it keeps a special SAS session running waiting for the coming requests. Using application server method alone can greatly shorten the response time.

Currently, the lack of utilities for quickly generating GUI components still is a main obstacle in development of multilevel SAS intranet application. Building those utility macros can speed up develop process in certain degrees, but non-visual SCL object combined with SAS/AF Frame entry alike visual construction tool are necessary for rapidly develop full scale SAS based intranet application.

RERFENCES

Tom Savola and Mark Brown, *Using Html: Special Edition*, Que Corp., 1996

Mohammed Kabir, *CGI Primer Plus for Windows*, Waite Group Press, 1996

SAS Institute Staffs, *Web Tools Developed by SAS Institute*, http://www.sas.com/rnd/web/intro .html, SAS Institute, 1997

David Shinn and John Hansen, *SAS Software and the Web: Creating a Common Gateway Interface*, Observations 1Q/97, SAS Institute, 1997

Michael Friendly, *sascgi: A SAS - WWW Gateway*, http://www.math.yorku.ca/SCS/Online/sascgi/index .html, 1997

SAS Institute Staffs, *Running SAS Graphics Procedures and other SAS Procedures on the Web,* http://www.sas.com/techsup/download/ web/misc/sample5/read.txt, SAS Institute, 1997

CONTACT

James Sun
Constat Systems
83 Regal Drive
Monmouth Junction, NJ 08852
internet: SUNJ@PFIZER.COM