No More Downloading - Using SAS/ODS to Create SAS Graphs and HTML Documents for OS/390 Systems

Patricia Wingfield, Bank of America, Richmond, VA

Abstract

With the advent of mainframe web server software, such as IBM's 'OS/390 HTTP Server', and with SAS Release 8's new ODS facility, it is now possible to create HTML documents in OS/390 data sets - PDS/E, sequential or HFS. Graphs and web pages stored in these OS/390 data sets can be viewed via a web browser. It is no longer necessary to download the documents to another type of platform for viewing on the internet or intranet. This paper covers SAS coding techniques that you should find helpful for developing HTML and SAS/GRAPH documents for storing and displaying on OS/390 systems.

Introduction

Company-wide intranets are a desirable way to distribute information regarding OS/390 system performance, capacity planning, application reporting etc. Some sites currently accomplish this by using SAS to create 'flat files' of statistics that are then downloaded to a PC. They can then be incorporated into a spreadsheet and put into a document for printing or intranet viewing. Other sites may create SAS graph files on OS/390 and then download them to a PC. Either way, this process tends to be manual, time consuming, and prone to error.

IBM's 'OS/390 HTTP Server' now enables OS/390 to become a web server itself. With technology like this, it is now possible to use SAS/ODS and SAS/GRAPH to create 'gif' graphics files and HTML documents in OS/390 data sets such as sequential files and PDS/E's (also HFS files) and to directly view them on the internet or company intranet. No conversion or translation is necessary, and of course no downloading to another platform needs to be done.

In this paper I will cover coding concepts and techniques that should help you to develop web pages that are created on, stored in, and viewed by OS/390 systems.

URLs and Links

The URL used to access an OS/390 data set via OS/390 HTTP Server is typically of the form:

http://lparname.company.com/MVSDS/'HIGHLEVL.HTML(REPORT)'

"Lparname" has to be the domain name of an lpar on which the 'HTTP Server' started task is actually running. "MVSDS" indicates to the HTTP Server that the following path name is that of an OS/390 data set. Next, are the data set and member names that you want to access, enclosed in single quotes.

Generally, no changes (i.e. parameters changes etc.) need to be made to the HTTP Server in order to access any OS/390 data set. 'MVSDS' is a GWAPI plug-in to the HTTP Server that is supplied by IBM that enables you to access an MVS data set as a static web page. You must, however, ensure that the HTTP Server started task has RACF 'READ' access to any data sets that you will be accessing.

The first part of the URL (up to and including the 'MVSDS/') is the same as long as your data sets are all on the same OS/390 node. Therefore, the only part of the URL that you have to specify to SAS in your programming is the very last pathname, i.e. the data set name and the member name, enclosed in single quotes.

There are two types of links that you have to specify to SAS – the first one is a pointer from one HTML document (i.e. PDS/E member) to another. This type of link is pointed to in the generated HTML code by an HTML 'HREF=' parameter or, in a frame, a 'FRAME SRC='. The second type of link is a link between an HTML document and a graphics 'gif' file created by a SAS graphing procedure. This type of link is pointed to by a 'IMG SRC=' HTML parameter.

Anchor points are specific locations within an HTML document. When you use SAS procedures with a BY variable, the HTML code that is generated will contain multiple anchor points – one for each occurrence of the BY variable. Code that points to an anchor point must contain the path name (i.e. data set name + member name in single quotes) concatenated to the anchor point name. You can see the anchor points in the generated HTML code. They look like:

```
<A NAME="ANCHOR1"> </A>
```

Sample code to generate some HTML code and the links between them is:

```
ODS HTML
BODY='CPUA'(URL="'HIGHLEVL.HTML(CPUA)'")
CONTENTS='CONCPUA'
         (URL="'HIGHLEVL.HTML(CONCPUA)'")
FRAME='FRMCPUA'
         (URL="'HIGHLEVL.HTML(FRMCPUA)'")
ANCHOR='ANCHOR'
GPATH=ODSGRAPH(URL="'HIGHLEVL.GIF(CPUA)'");

PROC GPLOT DATA=CPU;
  PLOT CPUPCTBY * DATE /
       HAXIS = AXIS1
       HZERO
       NAME=CPUA
       VAXIS = AXIS2
       VZERO;
    BY SYSTEM;
RUN;
```

When accessing the web pages generated by the above code, the URL to specify is:

http://lparname.company.com/MVSDS/'HIGHLEVL.HTML(FRMCPUA)

A table of contents, with one entry for each system graphed is displayed by your web browser on the left. The graphs are displayed on the right. Selecting a particular system will bring you directly to that sytem's graph. The code above generates three members in data set 'HIGHLEVL.HTML'. The first is the frame member, 'FRMCPUA', which will contain "FRAME SRC=" parameters to point to the 'CONTENTS' and 'BODY' members. The HTML code generated in 'FRMCPUA' looks like this:

```
<FRAME MARGINWIDTH="4" MARGINHEIGHT="0"
SRC="'HIGHLEVL.HTML(CONCPUA)'" NAME="contents"
SCROLLING=auto>
<FRAME MARGINWIDTH="9" MARGINHEIGHT="0"
SRC="'HIGHLEVL.HTML(CPUA)'" NAME="body"
SCROLLING=auto>
```

The second member is the contents member, 'CONCPUA', which contains one HREF= pointer to the 'body' member, or CPUA, of the HTML code for each BY variable. An anchor point is concatenated by SAS to each pointer. A pointer from 'CONCPUA' to an anchor point in the 'CPUA' member that is for the sixth occurrence of the BY variable looks like:

```
<A HREF="'HIGHLEVL.HTML(CPUA)'#ANCHOR5"
TARGET="body">
```

The third member is the 'body' member itself – 'CPUA'. This contains the bulk of the HTML code and any pointers to the graphics files in the '.GIF' data set. A pointer to a graph looks like this:

```
<IMG SRC="'HIGHLEVL.GIF(CPUA)'" border="0"
USEMAP="#idx_map">
```

Actually, the pointers to the 'gif' PDS members is the one case where the SAS generated code does not come out quite right. I will discuss that further in section "GIF Pointers".

Finally, one graph member for each system is created in data set 'HIGHLEVL.GIF'. The PDS member name specified in the GPATH URL parameter should be the same as the NAME= parameter in the PLOT statement. In this example, one CPUAn member is created for each system, where n = null to (# of systems – 1).

Diagramming these relationships, we have:

```
FRAME→ CONTENTS(HREF=)→BODY(SRC=)→GRAPHS
     → BODY
```

TEMPLATE

You can use PROC TEMPLATE to modify the default SAS ODS Templates. Sample code to do this is:

```
PROC TEMPLATE;
  DEFINE STYLE CPEP1;
    PARENT=STYLES.DEFAULT;
    NOTES 'USED FOR HTML FRAMES';
REPLACE fonts
    "Fonts used for CPEP1" /
    'TitleFont2' = ("Arial, Times",3,Bold
Italic)
    'TitleFont' = ("Arial, Times",4,Bold
Italic)
    'StrongFont' = ("Arial, Times",3,Bold)
    'EmphasisFont' = ("Arial,
Times",2,Italic)
    'FixedEmphasisFont' = ("Arial",1,Italic)
    'FixedStrongFont' = ("Arial",1,Bold)
    'FixedHeadingFont' = ("Arial",1)
'FixedHeadingFont' = ("Arial",1)
    'BatchFixedFont' = ("SAS Monospace,
Times",1)
    'FixedFont' = ("Arial",1)
    'headingEmphasisFont' = ("Arial,
Times",3,Bold Italic)
    'headingFont' = ("Arial, Times",3,Bold)
    'docFont' = ("Arial, Times",2);
 REPLACE color_list
    "Colors used for CPEP1" /
    'fgB2' = cx003399
    'fgB1' = cxFF0000
    'bgA3' = white
    'fgA2' = black
    'bgA2' = white
    'fgA1' = black
    'bgA1' = white
```

The above code changes the default fonts and colors.

Data Sets

One of the qualifiers of the data set in which your SAS generated HTML code resides ('HIGHLEVL.HTML') should be 'HTML', in order that the document be interpreted correctly as HTML code instead of text. If you are accessing a data set that contains graphics (gif files), then the data set must contain 'GIF' as one of the qualifiers, in order for your browser to display it as a graphic. If the data set name does not contain 'HTML' or 'GIF' as a qualifier, then your browser will present it as simple text.

I use PDS/E's in all of my work, without any problems. Because of the requirement that 'HTML' and 'GIF' be qualifiers in the html and graphics data set names, respectively, you will have to allocate separate data sets for these two types of data. Sample code to allocate your files is:

```
FILENAME ODSHTML "HIGHLEVL.HTML"
                 DSNTYPE=LIBRARY
                 DSORG=PO
                 RECFM=VB
                 LRECL=512
                 BLKSIZE=27998
                 DISP=(NEW,CATLG,DELETE)
                 SPACE=(CYL,1,1,1));

FILENAME ODSGRAPH "HIGHLEVL.GIF"
                 DSNTYPE=LIBRARY
                 DSORG=PO
                 RECFM=VB
                 LRECL=512
                 BLKSIZE=27998
                 DISP=(NEW,CATLG,DELETE)
                 SPACE=(CYL,1,1,1));
```

Be very sure to have a record length of at least 512. A short record length can create errors in the generated HTML code. I initially had my record length set to '80', and had some of the HTML code generated incorrectly, without any SAS error messages being issued. At 512, I haven't had this problem.

GOPTIONS

A sample GOPTIONS statement to use with ODS HTML is:

```
GOPTIONS DEVICE=GIF TRANSPARENCY NOBORDER
         RESET=GLOBAL GUNIT=PCT BORDER
CBACK=WHITE
         COLORS=(BLACK RED BLUE GREEN PURPLE
ORANGE) CTEXT=BLACK
         HTITLE=4 HTEXT=2 FTITLE=SWISSB
FTEXT=SWISSB
         XPIXELS=700 YPIXELS=390;
```

ODS Statements

Sample ODS statements to use for creating HTML documents on OS/390 are:

```
ODS TRACE ON;
ODS LISTING CLOSE ;
ODS HTML
BODY='CPUA'(URL="'HIGHLEVL.HTML(CPUA)'")
         RECORD_SEPERATOR=NONE
         PATH=ODSHTML

GPATH=ODSGRAPH(URL="'HIGHLEVL.GIF(CPUA)'")
         STYLE=CPEP1;
```

The TRACE statement is convenient during testing. The RECORD SEPERATOR=NONE is strongly recommended. It prevents the HTML records from wrapping. The HTML document will still work the same without this parameter, but it is very difficult to examine it for debugging purposes, with the records all strung together. The PATH and GPATH parameters point to the filerefs for the HTML and GIF data sets, respectively. The absence of the CONTENTS and FRAME parameters indicate that this HTML statement will only be used for a procedure without a BY statement.

Drill Downs

In order to drill down from graph points or graph legends, the HREF= parameter that points to the path (i.e. data set name + member name + anchor point) of the HTML location that you want to drill down to must be supplied as a variable in the data set being graphed. For OS/390, this means that you have to supply, in the SAS data set to be graphed, a variable of the form:

```
HREF="'HIGHLEVL.HTML(CPUATAB)'#ANCHOR"
```

'HIGHLEVL.HTML(CPUATAB)' is the data set name and member name that contain the HTML code that you want to drill down to. In this case, since our example is not for a procedure with a BY variable, there is only one anchor point in the document – '#ANCHOR'.

Here is example code to create a graph of average CPU busy values by month for one system. You want to be able to drill down from any point on the graph to a table created by PROC TABULATE of all the values for that system.

```
DATA MTHCPU(KEEP=SYSTEM DATE CPUBUSY
HREFRPT);
  LENGTH HREFRPT $56.;
  SET MTHCPU;
  HREFRPT =
"HREF=""'HIGHLEVL.HTML(CPUATAB)'"||
            "#ANCHOR" || '"';
RUN;

FILENAME ODSHTML "HIGHLEVL.HTML" DISP=SHR;
FILENAME ODSGRAPH "HIGHLEVL.GIF" DISP=SHR;

GOPTIONS DEVICE=GIF TRANSPARENCY NOBORDER
```

```
        RESET=GLOBAL GUNIT=PCT BORDER
CBACK=WHITE
        COLORS=(BLACK RED BLUE GREEN PURPLE
ORANGE) CTEXT=BLACK
        HTITLE=4 HTEXT=2 FTITLE=SWISSB
FTEXT=SWISSB
        XPIXELS=700 YPIXELS=390;

ODS TRACE ON;
ODS LISTING CLOSE ;

ODS HTML
    BODY='CPUA'
         (URL="'HIGHLEVL.HTML(CPUA)'")
    RECORD_SEPERATOR=NONE
    PATH=ODSHTML
    GPATH=ODSGRAPH
         (URL="'HIGHLEVL.GIF(CPUA)'")
         STYLE=CPEP1;

PROC GPLOT DATA=MTHCPU;
   PLOT CPUBUSY * DATE  /
        HAXIS = AXIS1
        HZERO
        VAXIS = AXIS2
        VZERO
        NAME= 'CPUA'
        HTML=HREFRPT;

ODS HTML BODY='CPUATAB'
        (URL="'HIGHLEVL.HTML(CPUATAB)'")
        ANCHOR='ANCHOR'
        RECORD_SEPERATOR=NONE
        PATH=ODSHTML
        STYLE=CPEP1;

PROC TABULATE DATA=MTHCPU
       STYLE=[just=center font_size=1.5];
     CLASS DATE;
     CLASSLEV  DATE /
           STYLE = [font_size=2
           font_weight=bold];
     VAR CPUBUSY;
         TABLE  DATE, CPUBUSY='Total CPU
    %Busy'*MEAN=' '*F=5.1;
RUN;
```

The HTML=HREFRPT of the PROC GPLOT tells it which variable name contains the link to the drill down. In the prior DATA step, variable HREFRPT was filled in with the PDS/E name, member name, and anchor name of the HTML code that will contain the PROC TABULATE results. SAS takes this path name that you supply and plugs it into the HTML 'MAP' statement that describes, for each point in the graph, what the appropriate drill down is to.

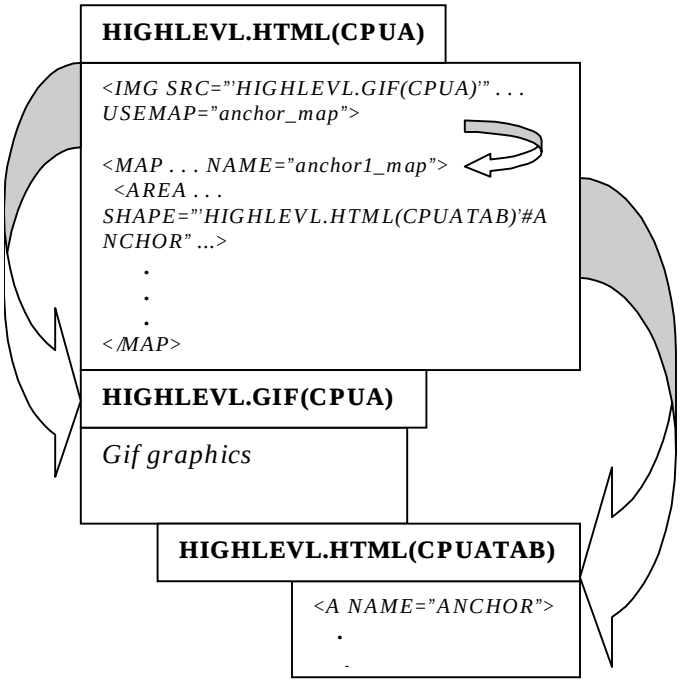
After this code is run, there are two members in the 'HIGHLEVL.HTML' PDS/E, 'CPUA' and 'CPUATAB'. There is one member in the 'HIGHLEVL.GIF' PDS, 'CPUA'.

When accessing the following URL:

http://lparname.company.com/MVSDA/'HIGHLEVL.HTML(CPUA)'

a graph showing the 'CPU Busy' values is displayed. Pointing and clicking on any point in the graph will bring you to HTML document 'CPUATAB', PROC TABULATE of the values of the graph.

Following is a graphical representation of the PDS/E members created and the internal HTML links between them:



The same thing can be done for a data with a BY variable. The main difference is that the CONTENTS and FRAME parameters should be specified for the ODS HTML statement; and the link variable anchor point has to be incremented for each occurrence of the variable. Here is an example of a data step to do that:

```
DATA MTHCPU (KEEP=SYSTEM DATE CPUBUSY
HREFRPT);
  RETAIN SYSSAVE '                    '
         CTR -1;
  INFORMAT CTR  2.
           SYSSAVE $20.;
LENGTH HREFRPT $58.;
  SET MTHDSK;
  IF SYSSAVE LT SYSTEM THEN
    DO;
      SYSSAVE = SYSTEM;
      CTR = CTR + 1;
    END;
  HREFRPT =
'HREF="''HIGHLEVL.HTML(CPUATAB)''' ||
'#ANCHOR' ||
            PUT(CTR,Z2.0) || '"';
  HREFRPT =
TRANWRD(HREFRPT,'#ANCHOR00','#ANCHOR');
  HREFRPT =
TRANWRD(HREFRPT,'#ANCHOR0','#ANCHOR');
RUN;
```

PROC REPORT

Two of the most powerful aspects of the ODS HTML features of PROC REPORT are the ability to modify the attributes of a particular cell in a table, dependent on the value of that cell, and the ability to turn the contents of any cell in a table into a URL. In the following sample code, which illustrates how to do this, field 'CPUTOT' is highlighted if it is over a threshold value (CPUTOTOP). In either case, it is also a URL, which points to a graph of CPU per cent busy values.

```
ODS HTML
BODY='EXCEPT'(URL="'HIGHLEVL.HTML(EXCEPT)'")
       RECORD_SEPERATOR=NONE
       PATH=ODSHTML
       STYLE=CPEP1;

PROC REPORT DATA=EXCEPT NOWD HEADLINE
SPLIT='*' LIST
      style(report)=[outputwidth=100
                     cellspacing=1
                     background=black
                     borderwidth=2];

COLUMN URLCPUA('Cpu % Busy' CPUTOTOP
CPUTOT);

DEFINE URLCPUA / DISPLAY NOPRINT;

DEFINE CPUTOTOP / DISPLAY
        STYLE(HEADER)=[font_size=1.5];

DEFINE CPUTOT / DISPLAY
        STYLE(HEADER)=[font_size=1.5];

COMPUTE CPUTOT;
   CALL DEFINE(_COL_, 'URL', URLCPUA);
   CALL DEFINE(_COL_, "STYLE",
      "STYLE=[flyover='Total CPU Trends' ]");
    IF CPUTOT > CPUTOTOP THEN
       DO;
         CALL DEFINE(_COL_, "STYLE",
          "STYLE=[font_weight=bold
           background=CXCCCCCC
           flyover='Total CPU' '
           font_style=italic]");
       END;
ENDCOMP;
RUN;
```

In the data step that created SAS data set 'EXCEPT', that is input to the PROC REPORT, field URLCPUA had been set to a character variable containing the literal, 'HIGHLEVL.HTML(CPUA)'. This OS/390 PDS/E member contains HTML code produced by PROC GPLOT. Clicking on the field 'CPUTOT' in the web page produced by this PROC REPORT will display the graphics web page.

GIF Pointers

As mentioned previously, the one type of link that does not come out correctly in the generated SAS HTML code are pointers to graphic, or 'gif' files. This is because SAS insists on appending the suffix '.gif' to the URL in the 'SRC=' parameter which points to the graphic. This is fine on most platforms, and if will work in OS/390 if you are using HFS files. However, it doesn't work with PDS/E's! In order to resolve this problem, I have written both a REXX edit macro and a SAS program. I use the REXX edit macro in a testing situation, and the SAS program to go against the generated HTML code in batch, which is useful for scheduled production.

If your ODS statement for a subsequent PROC GPLOT is:

```
ODS HTML
BODY='CPUA'(URL="'HIGHLEVL.HTML(CPUA)'")
RECORD_SEPERATOR=NONE
          PATH=ODSHTML
   PATH=ODSGRAPH(URL="'HIGHLEVL.GIF(CPUA)'")
     STYLE=CPEP1;
```

Then SAS will generate HTML code to point to the 'gif' PDS members that looks like:

```
SRC="'HIGHLEV.GIF(CPUA)'cpuaN.gif"
```

This needs to be changed to:

```
SRC="'HIGHLEVL.(CPUAN)'"
```

A REXX edit macro that can do this is:

```
SUBCOM ISREDIT; IF RC=1 THEN;
  DO;
    SAY 'YOU ARE NOT IN EDIT MODE';
    EXIT 8;
  END;
ADDRESS ISREDIT;
IF ADDRESS() /= 'ISREDIT' THEN;
  DO;
    SAY ADDRESS();
    SAY 'YOU MAY NOT EXECUTE INITS UNLESS IN
ISPF EDIT';
    EXIT 8;
  END;
'MACRO (GIFPTR)';
'CURSOR = .ZFIRST 1';
'RESET EXCLUDED';
UCGIFPTR=TRANSLATE(GIFPTR);
LCGIFPTR=TRANSLATE(UCGIFPTR,'abcdefghijklmno
pqrstuvwxyzZ0123456789','ABC
DEFGHIJKLMNOPQRSTUVWXYZ0123456789');
"CHANGE "||UCGIFPTR||")'"||LCGIFPTR||"
"||UCGIFPTR||" ALL";
 IF RC \= 0 THEN SIGNAL ERROR1;
'CURSOR = .ZFIRST 1';
"CHANGE .gif" || '"' || " )'" || '" ALL';
IF RC \= 0 THEN SIGNAL ERROR1;
ADDRESS ISPEXEC;
ZEDSMSG='Graphics links changed';
ZEDLMSG='Links now of the form DSN(CPUDn)';
'SETMSG MSG(ISRZ000)';
EXIT 0;
ERROR1:
ADDRESS ISPEXEC;
ZEDSMSG='Error on change command';
ZEDLMSG='Edit links manually';
 'SETMSG MSG(ISRZ000)';
 EXIT 8;
```

To invoke the macro: place the above edit macro into your SYSPROC concatenation; get into edit for the HTML member that has to be changed (in this example 'HIGHLEVL.HTML(CPUA)', which is the ODS HTML BODY member; and enter 'GIFPTR XXXX', where 'XXXX' is the name that you specified for the PDS member name in your GPATH statement and NAME= parameter.

To change the links in batch, simply run the following SAS program against the generated SAS BODY member code:

```
OPTIONS SOURCE SOURCE2;
DATA _NULL_;
  LENGTH FILENAME $50.
         GIF
         GIFLC    $4.;
  INFILE HTMLIN FILENAME=FILENAME;
  FILENAME=TRIM(FILENAME
MEMPOS=INDEX(FILENAME,'(');
MEMPOS=MEMPOS+1;
  GIF=SUBSTR(FILENAME,MEMPOS,4);
CALL SYMPUT('GIF',GIF);
GIFLC = LOWCASE(GIF);
CALL SYMPUT('GIFLC',GIFLC);
RUN;
DATA _NULL_;
  LENGTH GIFPTR $23.;
  INFILE HTMLIN LENGTH=VBLEN COLUMN=COLUMN
MISSOVER;
  FILE   HTMLOUT;
INPUT @".GIF(&GIF)'" X $1. @;
IF MISSING(X) THEN
     DO;
       PUT _INFILE_;
       INPUT;
       RETURN;
     END;
   COLPTR = COLUMN - 7;
COLREST = COLUMN;
RESTLEN = VBLEN-COLREST+1;
INPUT @1       BEGIN   $VARYING512. VBLEN
      @COLPTR  GIFPTR  $18.
      @COLREST REST    $VARYING472. RESTLEN;
   GIFPTR = TRANWRD(GIFPTR,")'&GIFLC",'#');
GIFPTR = TRANWRD(GIFPTR,"&GIF#","&GIF
GIFPTR = TRANWRD(GIFPTR,".gif""",")'""");
QUOTE=INDEXC(GIFPTR,'"'
GIFPTR=SUBSTR(GIFPTR,1,QUOTE+1);
GIFPTR = TRIM(GIFPTR);
BEGIN=SUBSTR(BEGIN,1,COLPTR-1);
SPACEPOS=INDEX(REST,' ');
REST=SUBSTR(REST,SPACEPOS);
NEWLINE=TRIM(BEGIN)||TRIM(GIFPTR)||'
'||TRIM(REST);
PUT @1 NEWLINE;
RETURN; RUN;
```

Testing & Debugging Hints and Tips

Of course, the way to test all of the above is to just get into your favorite inter/intranet browser, type in the URL, and hit enter! When you make a change, and want to see it immediately, hit the 'reload' or 'refresh' button on your browser.

I have found that when I can't figure out why a web page is not acting the way I think it should be, it sometimes helps to look at the HTML generated code. The problem is often with a link, and sometimes you are not really specifying the link the way that you think you are (i.e. superfluous spaces etc.) Looking directly at the generated HTML can often clear up the problem. You don't have to know a lot of HTML, and a beginner's level reference book helps a lot.

I found the SAS/ODS features of SAS Release 8.1 to be very reliable and robust. The biggest problems that I had were those involved in figuring out how to establish my links etc. in a specifically OS/390 PDS/E environment.

A very useful 'by-product' of OS/390 HTTP Server is that it provides a quick and easy way to get printouts of your source code, JCL etc. Just type in the data set name in the appropriate URL, select on the member name, and then use the FILE/PRINT facility of your browser. You only have to be sure that the HTTP Server started task has RACF READ access to the data sets you are accessing.

JAVA

We have recently done some work with JAVA. To use java graphics instead of gif files, replace the 'DEVICE=GIF' on your 'GOPTIONS' with 'DEVICE=JAVA'. You have to download the SAS java applets from SAS, and then upload them to an hfs file on your OS/390 system. This hfs file must be specified as a CLASSPATH to your OS/390 HTTP Server. You then also specify in your ODS HTML statement:

```
 Attributes = ("Codebase"
="http://lparname.companyname.com/sas/javaapplets")
       Archive  = "graphapp.jar";
```

The java code and data all gets included in your HTML pages, there are no gif files.

Advantages that we have found to this method are that we don't have to have two data sets for graphics, and so of course don't have to worry about the 'gifptr' problem. Also the graphics appear crisper, and can be dynamically modified once they are displayed. There is also a very nice box that pops up over all the graph points that displays the graph values, and drill downs are automatic.

Disadvantages are that if you have a lot of data composing a graph, page download time can be prohibitive. Also 'plots' don't come out nearly as well as 'charts', and drill downs are not necessarily what you want them to be. However, they can be overridden by specifically coding a drill down. Finally, this facility is not fully documented yet, although information on it is available on the SAS web site.

Conclusion

The new SAS/ODS facility, coupled with OS/390 web software such as OS/390 HTTP Server, greatly expands the usefulness of SAS for OS/390 reporting. OS/390 programmers can now put their web-based applications in OS/390 PDS's and PDS/E's. All of the powerful reliability, availability and security features of OS/390 are now available for this type of internet/intranet reporting.

Contact Information

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

Patricia Wingfield
Bank of America
8001 Villa Park Drive, C1-3
Richmond, VA 23228-6505
(804)-553-6218
pat.wingfield@bankofamerica.com