

No More Downloading - Using SAS/ODS® to Create Graphs and HTML Documents for z/OS Systems - Updated for SAS® v9.1

Patricia Wingfield, Bank of America, Richmond, VA

Abstract

With the advent of mainframe web server software, such as IBM's 'z/OS HTTP Server', and with the SAS ODS facility, it is possible to create HTML documents in z/OS data sets – pds/e, sequential or HFS. Graphs and web pages stored in these z/OS 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 created, stored and displayed on z/OS systems using SAS. This is a major update to a paper previously presented at NESUG, dealing with changes pertinent to SAS V9.

Introduction

Company-wide intranets are a desirable way to distribute information regarding z/OS 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 z/OS and then download them to a PC. Either way, this process tends to be manual, time consuming, and prone to error.

IBM's 'z/OS HTTP Server' enables z/OS to become a web server itself. With technology like this, it is possible to use SAS/ODS and SAS/GRAPH to create graphics files and HTML documents in z/OS 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 z/OS systems.

This paper is based on SAS release 9.1.

URLs and Links

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

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

"Lparname" is 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 a z/OS 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 z/OS 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.

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 z/OS 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.

An HTML report can be a one element report – i.e. either a flat file or one pds/e member. In this case, the report is self contained in the one pds/e member or flat file and there are no internal links between elements of the report. However, an HTML report can also be a multi-element report, produced when a reporting procedure is invoked with a 'BY' variable, or outputs from procedures are stacked into one report. In this case, the report must be in a pds/e. There are frame, contents, and one or more body members, each of which

is a separate member of the pds/e, and connected by internal HTML links. These tie the report together into a cohesive unit.

There are two types of links that SAS uses to hold a multi-element report together. The first one is a pointer from one HTML element (i.e. pds/e member) to another. This type of link is pointed to in the generated HTML code by an HTML 'HREF=' parameter. 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 an '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 should 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">&nbsp;</A>
```

Previously to SAS V9.1, it was necessary to post-process the links created by SAS ODS in order for them to correctly link the report elements together in a z/OS pds/e. There is now an ODS tagset, ODS TAGSET.MVSHTML, which produces correct z/OS links without post processing. Sample code to generate some HTML code and the links between them is:

```
ODS TAGSETS.MVSHTML
PATH="HIGHLEVL.HTML"
GPATH="HIGHLEVL.GIF"
BASE="http://lparname.company.com/MVSDS/"
BODY="CPU"
CONTENTS="CONCPU"
FRAME="FRMCPU"
(TITLE="CPU REPORT")
RECORD_SEPERATOR=NONE
NEWFILE=OUTPUT
STYLE=BOATOC1;

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

ODS TAGSETS.MVSHTML CLOSE;
```

When accessing the web pages generated by the above code, the URL to specify is:
[The frame member, 'FRMCPU', pointed to be the URL, contains "HREF=" HTML internal link parameters that point to the 'CONCPU' and 'CPU' members. A table of contents, the 'CONCPU' member, with one entry for each 'BY' occurrence, is displayed by your web browser on the left. The graphs and/or HTML 'CPU' members are the 'body' members displayed on the right. Selecting a particular 'BY' occurrence in the contents will bring you directly to that graph or HTML BODY document.](http://lparname.company.com/MVSDS/'HIGHLEVL.HTML(FRMCPU)'>http://lparname.company.com/MVSDS/'HIGHLEVL.HTML(FRMCPU)'</p></div><div data-bbox=)

The code above generates at least three members in data set 'HIGHLEVL.HTML'. The HTML code generated in 'FRMCPU' looks like this:

```

<li><a
href="http://lparname.companyname.com/MVSDS/'HIGHLEVEL.HTML (CONCPU) '">The
Table of Contents</a></li>
<li><a href="http://lparname.companyname.com/MVSDS/'HIGHLEVEL.HTML (CPU) '">The
Contents</a></li>
</ul>

```

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

```

<a
href="http://lparname.companyname.com/MVSDS/'HIGHLEVEL.HTML (CPU3) '#ANCHOR3"
target="body">

```

The next members are the 'body' members themselves - 'CPUn'. They contain the bulk of the HTML code and any pointers to the graphics files in the '.GIF' data set, if you are doing gif graphing. A pointer to a graph looks like this:

```



```

SAS numbers the first BODY member with no suffix, and then starts numbering them sequentially, starting with '1'.

Finally, one graph member for each system is created in data set 'HIGHLEVEL.GIF'. The pds/e member name specified in the GPATH URL parameter should be the same as the NAME= parameter in the PLOT statement. In this example, one CPUn 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

```

PROC TEMPLATE

SAS supplied some changes to PROC TEMPLATE to go with the TAGSET.MVSHTML for V9.1:

```

proc template;
edit tagsets.mvshtml;
define event listclass;
putq " class=" HTMLCLASS;
end;
define event margins;
put " text-indent: " indent;
put " text-indent: " indent;
put ";" NL / exists( indent);
put ";" NL / exists( indent);
put " margin-left: " LEFTMARGIN;
put ";" NL / exists( LEFTMARGIN);
put " margin-right: " RIGHTMARGIN;
put ";" NL / exists( RIGHTMARGIN);
put " margin-top: " TOPMARGIN;
put ";" NL / exists( TOPMARGIN);
put " margin-bottom: " BOTTOMMARGIN;
put ";" NL / exists( BOTTOMMARGIN);

```

```

end;
define event list_entry;
start:
unset $first_proc;
set $first_proc "true" /if total_proc_count ne 1;
put "
" /if exists( $first_proc, $proc_name);
put "<li";
putq " class=" HTMLCLASS;
put ">" NL;
trigger do_link /if listentryanchor;
trigger do_list_value /if not listentryanchor;
finish:
put "</li>" NL;
put split; /* Space after the procedure */
put split; /* Second space after procedure */
end;
define event leaf;
break;
end;
end;
run;

```

These need to be added to your PROC TEMPLATE code.

Data Sets

One of the qualifiers of the data set in which your SAS generated HTML code resides ('HIGHLEVEL.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 should 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 may present it as simple text, or you may even get 'page not found' messages.

Pds/e's work very well for your HTML and graphics data sets. They do not have the problems inherent in old-style PDS's. 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 "HIGHLEVEL.HTML"
DSNTYPE=LIBRARY
DSORG=PO
RECFM=VB
LRECL=512
BLKSIZE=27998
DISP=(NEW,CATLG,DELETE)
SPACE=(CYL,1,1,1));

FILENAME ODSGRAPH "HIGHLEVEL.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 (i.e. '80') can cause HTML code to be truncated, without any SAS error messages being issued. At 512, this problem has not occurred.

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 one element HTML documents on z/OS are:

```
ODS TRACE ON;
ODS LISTING CLOSE ;
ODS HTML BODY='CPUA'
(URL="'HIGHLEVL.HTML(CPUA) '")
RECORD_SEPERATOR=NONE
PATH=ODSHTML
STYLE=BOATMPLT1;
```

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 should be supplied as a variable in the data set being graphed. For z/OS, this means that you have to supply, in the SAS data set to be graphed, a variable of the form:

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

'HIGHLEVL.HTML(CPU)' 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(CPUA) ' " |
"#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 TAGSETS.MVSHTML
PATH="HIGHLEVEL.HTML"
GPATH="HIGHLEVEL.GIF"
BASE="http://lparname.company.com/MVSDS/"
BODY="CPUA"
(TITLE="CPU REPORT")
RECORD_SEPERATOR=NONE
NEWFILE=OUTPUT
STYLE=BOABASE;

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

ODS HTML BODY='CPUTAB'
(URL="'HIGHLEVEL.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;
ODS TAGSETS.MVSHTML CLOSE;

```

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 'HIGHLEVEL.HTML' pds/e, 'CPU' and 'CPUTAB'. There is one member in the 'HIGHLEVEL.GIF' PDS, 'CPU'.

When accessing the following URL:

[a graph showing the 'CPU Busy' values is displayed. Pointing and clicking on any point in the graph will bring you to HTML document 'CPUTAB', PROC TABULATE of the values of the graph.](http://lparname.company.com/MVSDA/'HIGHLEVEL.HTML(CPU)'>http://lparname.company.com/MVSDA/'HIGHLEVEL.HTML(CPU)'</p>
</div>
<div data-bbox=)

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

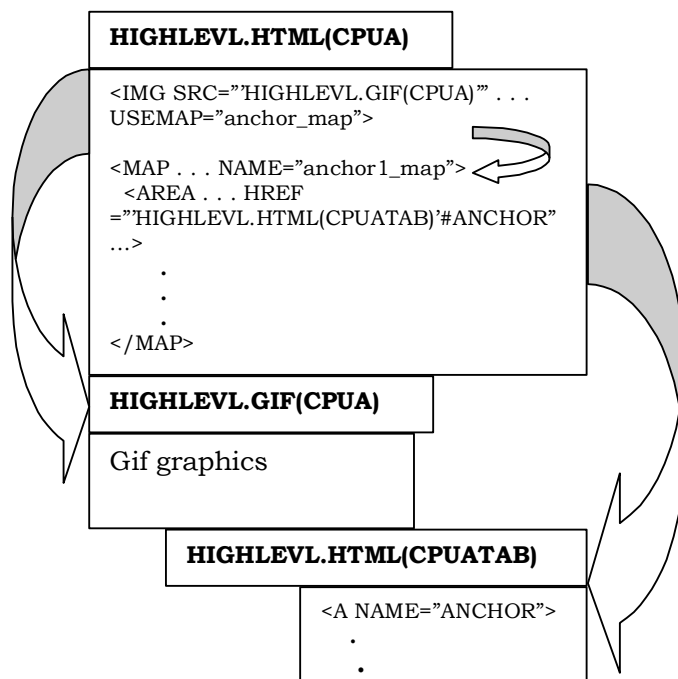


Figure 1. Internal HTML links schematic

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="'HIGHLEVEL.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=CXCCCCC
flyover='Total CPU' `
font_style=italic]");
END;
ENDCOMP;
RUN;
ODS HTML CLOSE;

```

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 z/OS 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.

Do not use the ODS TAGSET.MVSHTML if creating these types of drill downs. Instead use ODS HTML. The tagset will incorrectly modify the links inserted by the COMPUT/CALL DEFINE URL. Therefore, if you are creating a multi-element report, you will then have to post process your internal links in order for your report to work correctly. The SAS V8 version of this paper details how to post-process the links.

Converting to spreadsheets

HTML tabular output from PROC REPORT, TABULATE or PRINT can be easily made into spreadsheets WITHOUT having to resort to the creation of csv files. Some ways are:

You can click on a table; select edit/select all/copy, then copy into an empty spreadsheet. This way gives you a formatted spreadsheet, but you will need to do some minor alignment of titles etc.

Right click on a table and select 'Export to Microsoft Excel'. An empty spreadsheet will open up and in a few seconds the data, along with the titles will be displayed as an unformatted spreadsheet.

If the tabular report is a single element report, it can be directly downloaded to your pc and becomes a formatted spreadsheet (see 'Some Helpful HTTP Directives' below). Since the URL of multi-element reports point to the frame member, not the tabular report itself, this doesn't work for multi-element reports, unless you change your URL to directly point to the pds/e member that contains the tabular report.

Freeze headers

The following macro can be used after specifying your template. In this example 'BOA&TMPL' is the template being used and already specified.

```

%MACRO FREEZE(FRZE);
PROC TEMPLATE;
DEFINE STYLE STYLES.BOA&FRZE;
PARENT=BOA&TMPL;
STYLE TABLE FROM TABLE /
OUTPUTWIDTH=99%
PREHTML='<DIV
style="overflow:auto;
WIDTH:99%;HEIGHT:575PX"
id="data">'
posthtml="</DIV>";
STYLE HEADER FROM HEADER /
htmlstyle='z-index:20;
POSITION:relative;
TOP:
expression(document.getElementById("data").scrollTop-2)';
END;
RUN;
%LET TMPL=FRZE;
%MEND FREEZE;

```

This macro will ‘freeze’ your headers while scrolling down through tabular reports, making it much easier to keep track of what the data values are.

Java

You can also use java for your graphs. 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 z/OS system. This HFS file should be specified as a CLASSPATH to your z/OS HTTP Server. You then also specify in your ODS HTML statement:

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

Change the file names to your specific file names. 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 you don’t have to have two data sets for graphics. Also the graphics appear crisper, and can be dynamically modified once they are displayed. There is also a very nice flyover box that pops up over all the graph points that displays the graph values, and drill downs are automatic. The drill downs, however, may not be the type of drill down that you want. If so, the default drill down can be specifically overridden via the ‘HREF’ parameter as described under ‘Drilldowns’ previously in this paper.

Disadvantages are that if you have a lot of data composing a graph, page download time can be prohibitive. Also some chart options are ignored or overridden with java graphs, and there is no way to specify them. Finally, ‘PROC ANNOTATE’ does not always work as you would think and often does not work at all. In some cases, it does work, however, so you have to try it to see.

TAGSET vs. ODS HTML?

In some cases, SAS ODS HTML no longer works with V9.1, and you absolutely must use the TAGSET. This table summarizes when you to use the TAGSET or ODS HTML;

	ODS HTML	TAGSET
Single Element, without COMPUTED links	Yes	Yes
Single Element, with COMPUTED links	Yes	No - COMPUTED links will not work
Multi-element HTML only (no graphics) NEWFILE=NONE, no PROC REPORT COMPUTED links	Yes, although some unnecessary links generated in TOC - also can create very large report since one body member - long time to download	Yes, but can create very large reports in the one body member however - long time to download
Multi-element HTML only (no graphics) NEWFILE=OUTPUT, no PROC REPORT COMPUTED links	No - doesn’t work	Yes
Multi-element HTML only (no graphics) NEWFILE=OUTPUT, COMPUTED links	Yes, but will need to post-process links	No - COMPUTED links will not work
Multi-element HTML + gif NEWFILE=NONE (always use NEWFILE=NONE with gif)	Yes, although some unnecessary links generated and need to post process links.	Yes
Multi-element HTML + java graphs NEWFILE=OUTPUT (always use with java)	No - doesn’t work	Yes

The above table assumes that the PROC TEMPLATE changes detailed in the ‘TEMPLATE’ section above have been applied.

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.

When experiencing testing problems, it often helps to look at the generated HTML 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.

A very useful 'by-product' of z/OS 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.

Miscellaneous Helpful HTTP Directives

The 'AddType download' HTTP Server directive allows you to incorporate the ability to download MVS data sets from your generated web pages. To do this, have your HTTP Server administrator add the following directive to the HTTP Server:

```
AddType .DOWNLOAD text/download ebcdic 1.0 # MVSDS
```

The above directive will cause your browser to open the standard Windows download dialog box, whenever an MVSDS URL is selected that ends in '.DOWNLOAD'. You can therefore have a link in your web page pointing to an MVS data set that ends in '.DOWNLOAD'. The end user on the browser will be able to download this data set by clicking on the link in your web page.

The way that RACF security for MVS data sets is handled can also be changed via the 'UserId' directive. Setting the directive to 'UserId %%CLIENT%%', will cause a userid and password dialog box to appear the first time that a user enters an 'MVSDS' URL in their browser window. The user then can enter their TSO userid and password. Access to any URL on that particular HTTP server will then be allowed or denied to that user based on standard RACF security.

Conclusion

The new ODS TAGSET.MVSHTML, along with SAS ODS HTML, coupled with z/OS web software such as the z/OS HTTP Server, greatly expands the usefulness of SAS for z/OS reporting. z/OS programmers can put their web-based applications in z/OS flat files and pds/e's. All of the powerful reliability, availability and security features of z/OS 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
(804)-553-5768 (fax)
pat.wingfield@bankofamerica.com

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. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.