Whet Your Application Appetite with DSGI

Danielle L. Davis, SAS Institute Inc., Irvine, CA

ABSTRACT

With today's changing business needs, more applications are workstation based. The Data Step Graphical Interface (DSGI) toolset of SAS/GRAPH® Software is invaluable with today's demand for GUI based applications and presentation style reports. DSGI can be the tool of choice to easily print complex reports that exploit GUI features. With DSGI you can: 1) print text-based reports with mixed fonts and sizes without having to write a submit block in your Screen Control Language (SCL) 2) easily mix multiple reports and graphics on a single page 3) print hardcopy of SAS/AF® Software FRAME-based GUI screens that combine both text and graphics. This tutorial demonstrates how easily DSGI can be utilized to make these capabilities possible.

INTRODUCTION

Developing applications for today's sophisticated end - users require more than a menu-based system. GUI-based applications require the ability to create dynamic ad-hoc reports for any part of the application. End-users need the flexibility of printing out almost any view of their data that they are viewing on a particular screen. The SAS/GRAPH® Software feature DATA Step Graphics Interface (DSGI) gives you the ability to provide this functionality. Other SAS/GRAPH procedures could be used in some of the examples, but there are advantages to using DSGI to replace or integrate with other procedure output. Some of the advantages include the following:

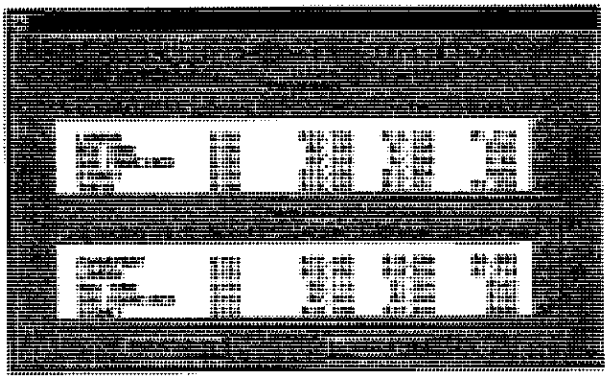
- Ability to write and compile DSGI with SCL without writing a submit block.
- Saving disk space
- Typically generates graphics faster that procedures that need annotation

This paper identifies some key areas where using DSGI can prove invaluable.

PRINTING A CUSTOM REPORT

Displaying a text-based report on a screen can be done very easily with SAS/AF Software. However, printing out the report can present some problems. If the report is in an extended table, a user may not be viewing all of the data on the screen but the print out should reflect all of the data. Another obstacle you may hit is when you are trying to create a WYSIWYG report including mixed fonts and sizes.

Lets take an example of the following screen:



Display 1

This example shows a report in an extended table with mixed fonts. When the user selects PRINT you would like the report to print all of the data and the mixed font. DSGI would allow you to do this without having to call any procedures. The following will execute when the user chooses the PRINT button:

```
PRINT:
      /* Set the printing device */
   rc=gset('DEVICE','WINPRTC');

/* Initialize DSGI Environment */
   rc=ginit();

/* Determine the window size to work with    */
/* This is dependent on the graphics device */
/* llx= min x dimension                      */
/* lly= min y dimension                      */
/* urx=max x dimension                       */
/* ury= max y dimension                      */
   call gask('WINDOW',0,llx,lly,urx,ury,rc);

/* Clear Graphics catalog entry */
   rc=graph('CLEAR','TEXT');

/* Define title font, size, color and        */
/* alignment then print title                */
   rc=gset('TEXFONT','SWISS');
   rc=gset('COLREP',1,'BLACK');
   rc=gset('TEXCOLOR',1);
   rc=gset('TEXHEIGHT',3.0);
   rc=gset('TEXALIGN','CENTER','NORMAL');
   rc=gdraw('TEXT',50,ury-2,'Actual Vs. Budget
                    Summary Report');

/* Define label size, alignment, and size */
   rc=gset('TEXHEIGHT',2.3);
   rc=gset('TEXALIGN','LEFT','NORMAL');
   rc=gdraw('TEXT',3,ury-10,'Originating Hub');
   rc=gdraw('TEXT',30,ury-10,'Period');
   rc=gdraw('TEXT',52,ury-10,'Actual');
   rc=gdraw('TEXT',65,ury-10,'Budget');
   rc=gdraw('TEXT',77,ury-10,'Variance');
/* Set first line of data to print */
  line= ury-15;
  /* Loop through data set */
  do while(fetch(dsid)= 0);
```

```
/* define height, font, alignment    */
/* and size of data                  */
   rc=gset('TEXHEIGHT',2.0);
   rc=gset('TEXFONT','SWISSL');
   rc=gset('TEXALIGN','LEFT','NORMAL');

/* print data for report */
   rc=gdraw('TEXT',3,line,hub);
   rc=gset('TEXALIGN','RIGHT','NORMAL');
   rc=gdraw('TEXT',38,line,period);
   rc=gdraw('TEXT',60,line,
       put(Actual,dollar8.));
   rc=gdraw('TEXT',75,line,
       put(budget,dollar8.));
   rc=gdraw('TEXT',87,line,
       put(Variance,dollar8.));

/* decrement line counter */
    line= line-3;
  end;

/* Update output */
   rc=graph('UPDATE');

/* Close DSGI environment */
    rc=gterm();
return;
```

When this code is executed it will produce the following output.

ACTUAL/ BUDGET YEARLY REPORT

1990 Summary

Originating Hub	Period	Actual	Budget	Variance
Frankfurt	Q190	$2,504	$2,215	$288
London	Q190	$2,201	$2,210	$-9
New York	Q190	$235	$227	$8
San Francisco	Q190	$822	$859	$-38
Syndey	Q190	$1,496	$1,503	$-6
Tokyo	Q190	$2,073	$1,839	$239

1991 Summary

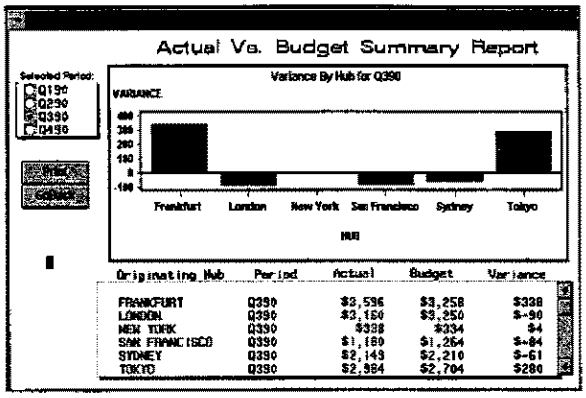
Originating Hub	Period	Actual	Budget	Variance
Frankfurt	Q191	$3,073	$2,732	$341
London	Q191	$2,701	$2,726	$-24
New York	Q191	$289	$280	$9
San Francisco	Q191	$822	$859	$-38
Syndey	Q191	$1,496	$1,503	$-6
Tokyo	Q191	$2,073	$1,839	$239

Output 1

This DSGI code may look long and tedious to program, however, the power to fully customize your output can make it worth the effort. DSGI can be completely data driven and can provide you with the power to develop customized reports that do not require you to write a submit block. The DSGI functions and routines will compile with the SCL code behind the frame. This allows you to have less overhead within the application.

MIXING GRAPHS AND REPORTS

SAS applications today also have the ability to allow a user to view a graph and a report or data on one screen. Therefore, users are requesting to print out this graph and report on one page. There are several SAS/GRAPH procedures that would allow you to place a report and a graph on one page. Let's take a look at Display 2 as an example of a FRAME entry.



Display 2

One method of generating a printout of this frame would be the following:

- Use PROC PRINT to create the report
- Use PROC GPRINT to save the report as graphics output to a graphics catalog.
- Use PROC GCHART to create the graph and save it to a graphics catalog
- Use PROC GREPLAY to create a template and place the graph and the report on one page.

This method can take up extra disk space, take more time to create the final result and the GOPTIONS tend to need adjusting to acount for any any skewness that the graph and report may acquire when placed in a template panel. This method also requires a large submit block of SAS language code.

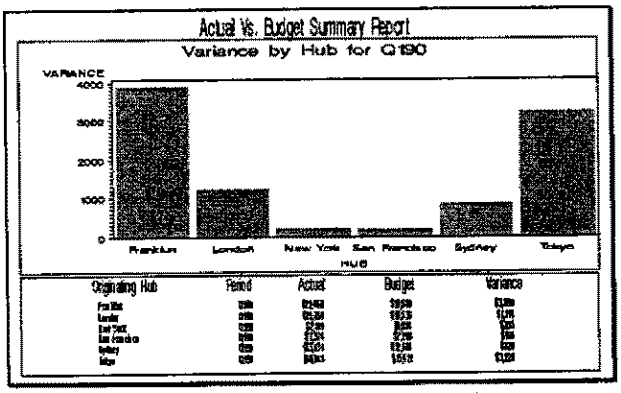
An alternate method is to use the GCHART procedure to create the chart and then utilize DSGI with SCL to create the report and finished output. This method allows the GCHART procedure to create the bar chart easily and then allow DSGI to incorporate the bar chart with the rest of the report. The advantage to using this method is the following:

- The submit block is processing only the code needed to create the bar chart and save it to a temporary graphics catalog.
- The report is created with DSGI functions that are using SCL to access the data rather than writing a DATA step.

Below is the DSGI code that would execute when the user chose to print:

```
PRINT:

  /* Create the Bar chart and save to the     */
  /* work temporary catalog                   */
  submit continue;
      goptions reset=all nodisplay dev=winprtc
               ftext=swiss htext=1.5 pct border
        cback=white ctext=black;
      pattern1 c=gray v=solid r=7;

  /* &selper = The radio box value */
      title h=3 pct 'Variance by Hub for
               &selper';
      proc gchart data= sasuser.iawper
                  gout=work.gseg;
           where period= '&selper';
          vbar hub /sumvar=variance frame
                      name='hubchrt' space=1;
        run;
        quit;

  /* Clear title stmt for DSGI graph */
       title;

  /* reset the DISPLAY and BORDER option */
       goptions display noborder;
        endsubmit;

  /* Define the printout device */
       rc=gset('DEVICE','WINPRTC');

  /* Initialize the DSGI environment          */
  /* and determine window size                */
       rc=ginit();
       call gask('WINDOW',0,llx,lly,urx,ury,rc);
       rc=graph('CLEAR','TEXT');

  /* Define title attributes for printed text */
       rc=gset('TEXFONT','SWISS');
       rc=gset('COLREP',1,'BLACK');
       rc=gset('TEXCOLOR',1);
       rc=gset('TEXHEIGHT',3.0);
       rc=gset('TEXALIGN','CENTER','NORMAL');
       rc=gdraw('TEXT',50,128,'Actual Vs.
                Budget Summary Report');

  /* Define attributes and print labels for   */
  /* the report                               */
       rc=gset('TEXHEIGHT',2.3);
       rc=gset('TEXALIGN','LEFT','NORMAL');
       rc=gdraw('TEXT',7,40,'Originating Hub');
       rc=gdraw('TEXT',35,40,'Period');
       rc=gdraw('TEXT',55,40,'Actual');
       rc=gdraw('TEXT',68,40,'Budget');
       rc=gdraw('TEXT',80,40,'Variance');

  /* create a box around report */
       rc=gdraw('BAR',5,13,93,43);

     /* Generate Report */
       rc=rewind(dsid);
       line= ury/2 -15;

  /* Fetch all data records for report */
     do while(fetch(dsid)= 0);
       rc=gset('TEXHEIGHT',1.6);
       rc=gset('TEXFONT','SWISSl');
       rc=gset('TEXALIGN','LEFT','NORMAL');
       rc=gdraw('TEXT',7,line,hub);
     rc=gset('TEXALIGN','RIGHT','NORMAL');
       rc=gdraw('TEXT',40,line,period);
       rc=gdraw('TEXT',63,line,
           put(Actual,dollar8.));
       rc=gdraw('TEXT',78,line,
           put(budget,dollar8.));
       rc=gdraw('TEXT',90,line,
           put(Variance,dollar8.));
       line= line-2;
     end;

  /* Define Viewport to place bar chart */
       rc=gset('VIEWPORT',1,.05,.40,.93,.90);
       rc=gset('WINDOW',1,0,0,100,100);
       rc=gset('transno',1);

  /* Insert Bar chart on page inside viewport */
    rc=graph('INSERT','hubchrt');

  /* Update and print then close DSGI         */
  /* environment                              */
       rc=graph('UPDATE','SHOW');
       rc=gterm();

  return;
```

Output 2 shows the results of the hardcopy printout.

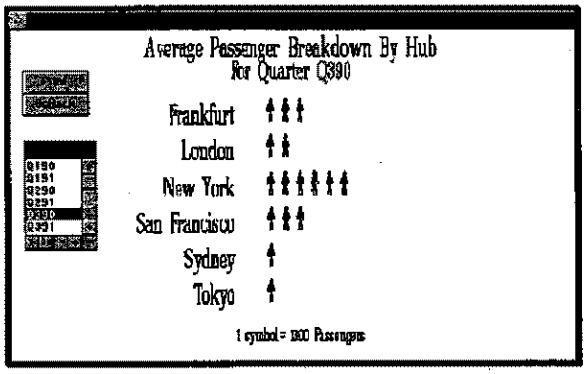


Output 2

With this method the report is not pushed into a small panel. Rather you have defined exactly where the report should start and the appropriate font size needed for the available space. You can also develop a generic Print method using this code as a template.

DYNAMIC CREATION OF CUSTOM REPORTS

Another example of how DSGI can provide additional functionality for developing applications is dynamically creating custom reports. Display 3 below is an example of how you can dynamically develop a custom report with DSGI.



Display 3

This frame dynamically creates the graph depending on the users selection of the period. There is no need to save a catalog of graphs for each quarter defined in the data. This saves disk space and allows the application to be much more data driven. Each time a new quarter is added to the data no application modifications are needed. This report could be created with a DATA step. However, doing it within SCL actually proves more valuable given the additional functionality of SCL. DSGI is also allowing you to create reports that cannot be easily created by SAS/GRAPH procedures. By dynamically creating this report you can:

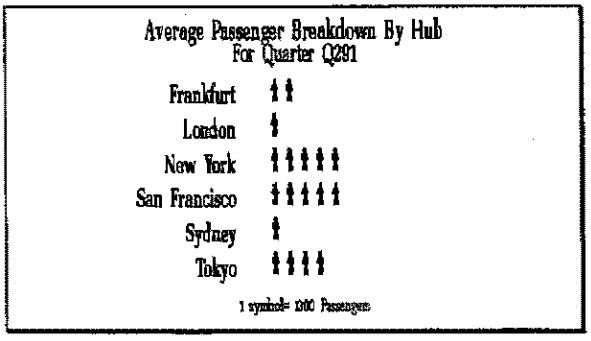
- Decrease the amount of disk space used
- Avoid storing each different scenario as a static GRSEG entry in a catalog.
- Allow the users easy printing of the report.
- Allow the ability to develop any type of custom report with mixed fonts and graphs.
- Ability to take full advantage of SCL functions

Below is the DSGI code that would generate this custom report.

```
length area $15 actual 8 budget 8 variance 8 btn
$6 avgpass 8 hub $15 period $5;

INIT:
  /* Open Data set */
  dsid=open('sasuser.iawpict','I');

  /* Calculate the Min & Max for # of    */
  /* passengers                          */
   rc=varstat(dsid,'AVGPASS','min max mean'
        ,minpass,maxpass,meanpass);
  call set(dsid);
  link PERIOD;
return;

PERIOD:
  /* Subset data based on user's selection */
  call notify('QUARTER','_GET_LAST_SEL_',indx,
              isel,period);
  rc=where(dsid,'PERIOD='||quote(period) );
  Link MKREPT;
return;

MKREPT:
   /* Calculate the number of unique Hubs */
   rc=varstat(dsid,'HUB','NUNIQUE',numhub);

   /* Delete old graph if it exists */
   rc=ginit();
   if cexist('WORK.GSEG.REPORT.GRSEG') then
      rc=graph('delete','REPORT');
   rc=gterm();

  /* Define output Device and Initial DSGI    */
  /* and Graphics area                        */
     rc=gset('DEVICE','IMGTIFB');
     rc=ginit();
     rc=graph('CLEAR','REPORT');
     call gask('WINDOW',0,llx,lly,urx,ury,rc);

   /* Calculate x and Y ranges defined        */
   /* by device                               */
     xrange= urx -llx;
     yrange= ury*.85 - lly;

     /* define attributes and draw titles */
     rc=gset('TEXFONT','ZAPFB');
     rc=gset('COLREP',2,'BLUE');
     rc=gset('TEXCOLOR',2);
     rc=gset('TEXHEIGHT',8.5);
     rc=gset('TEXALIGN','CENTER','NORMAL');
     rc=gdraw('TEXT',xrange/2,ury-8,'Average
              Passenger Breakdown By Hub');
     rc=gset('TEXHEIGHT',7.7);
     rc=gdraw('TEXT',xrange/2,ury-15,
              'For Quarter '||period);

   /* define Graphics Areas */
   /* X Labels - Hub Names */
      xlabel = .40 * xrange;
  /* min X dimension */
      xmin   = .45 * xrange;
  /* max X dimension */
      xmax   = .97 * xrange;
  /* min Y dimension */
      yaxmin = .20 * yrange;
  /* max Y dimension */
      yaxmax = .95 * yrange;
  /* Calculate symbol Height */
     ywidth = floor((yaxmax-yaxmin)/
                    (numhub + 1));

  /* calc. margins between rows of symbols */
     ymargn = ((yaxmax-yaxmin) -(ywidth*numhub))
                    / (numhub-1);

  /* Calculate X Min and Max for Axis range */
     vmin= round((minpass -100),100);
     vmax=(floor(maxpass/100)+1) *100;

  /* Calc. space between symbols in          */
  /* X direction                             */
     incr=(xmax-xmin)/ 20;

    /* Calculate # of passengers 1 symbol */
     symvalue=round(vmin,10);

    /* define attributes for legend text */
     rc=gset('TEXFONT','ZAPF');
     rc=gset('COLREP',1,'BLACK');
     rc=gset('TEXCOLOR',1);
     rc=gset('TEXHEIGHT',5.5);
```

```
  rc=gset('TEXALIGN','LEFT','NORMAL');
  /* Draw legend */
  rc=gdraw('TEXT', .4*xrange, .07*yrange,
     '1 symbol='||left(put(symvalue,4.)) || '
           Passengers ');

/* define start line for first hub and     */
/* it's symbols                            */
  stline= yaxmax;

/* Loop through all subset data */
do while(fetch(dsid)= 0);

  /* Define y value for row */
  ybar= stline -ywidth;

  /* define # of Symbols to be placed */
  numpeop=avgpass / symvalue;

  /* Set up text attributes and draw  */
  /* hub name                         */
  rc=gset('COLREP',1,'BLACK');
  rc=gset('TEXCOLOR',1);
  rc=gset('TEXHEIGHT',8.0);
  rc=gset('TEXFONT','ZAPFB');
  rc=gset('TEXALIGN','RIGHT','BASE  ');
  rc=gdraw('TEXT',xlabel,ybar,hub);

  /* Set up text attributes */
  rc=gset('COLREP',1,'BLUE');
  rc=gset('TEXCOLOR',1);
  rc=gset('TEXHEIGHT',8.5);
  rc=gset('TEXFONT','MARKER');
  rc=gset('TEXALIGN','CENTER','NORMAL');
  /* Draw defined # of symbols for the hub */

  do i=1 to numpeop;
    xval=xmin+ incr/2 +((i-1)*incr);
    rc=gdraw('TEXT',xval,ybar,'Q');
  end;

  /* Define next Y coordinate for next row */
  stline= ybar -ymargn;
end;

   /* Update graph and close DSGI */
  rc=graph('UPDATE','NOSHOW');
  rc=gterm();
  call notify('GRAPH','_UPDATE_');
return;
```

Output 3 shows a hardcopy printout of the custom report.



Output 3

For a better example of this type of report (know as a pictogram) please refer to *Drawing a Pictogram: An Adventure into the Data Step Graphics Interface*, Jade Walker and Edie Jefferys, Observations, Fourth Quarter 1992. This example also shows how DSGI can help you develop any type of unique report that a user may request.

CONCLUSION

Incorporating DSGI with SCL can help develop robust applications and can be worthwhile. DSGI allows you to print necessary reports needed for your application. DSGI functions and routines can be compiled with your SCL this allows you to:

- Provide faster processing for graphical reports and mixed font reports
- Reduce or eliminate the need to add a submit block to your SCL.
- Reduces the need to store preprocessed static output.

REFERENCES

SAS Institute Inc. (1990), SAS/GRAPH® Software, Reference, Version 6, First Edition, Cary,NC: SAS Institute Inc.

SAS Institute Inc. (1994), Observations, *Drawing a Pictogram: An Adventure into the DATA Step Graphics Interface*, Jade Walker and Edie Jefferys, Fourth Quarter 1992, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1994), Observations, *Creating Tables Using the DATA Step Graphics Interface (DSGI)*, Harvey Monder, Third Quarter 1995, Cary, NC: SAS Institute Inc.

ABOUT THE AUTHOR

Danielle L. Davis is a Applications Consultant for SAS Institute Inc. in the Professional Services Division. She can be reached at (619) 672-1860.