

You Did That Report in SAS®!?: The Power of the ODS PDF Destination

Pete Lund, Looking Glass Analytics, Olympia, WA

Abstract

The Output Delivery System (ODS) has been around since SAS® version 7 and yet many people still don't realize that they use it every day just to send results to the output window. They're still more amazed when they see that publication quality reports in PDF files can be created with SAS and ODS.

This paper explores a number of ODS options in general and, more specifically, their use in creating PDF output. We will cover ODS ESCAPECHAR, which allows for inline formatting of titles, footnotes and other text and new syntax for version 9.2; ODS LAYOUT, which lets you place output wherever you want it on the page - even output from more than one procedure, both text-based output and graphics; inline formatting in PROC REPORT; the new world of DATA_NULL_ reporting using the ODS object and more.

We'll work from real life examples and see how you can produce output that looks like it took hours to create.

Introduction

This paper is an update of a paper that was first presented at SUGI 31 in San Francisco in 2006. There have been some nice changes to ODS in version 9.2, which was released since that paper was first written. We'll still go a number of the same real world examples that demonstrate techniques that can be used with the PDF destination to make your reports crisper, cleaner and more useful than ever before. Some new examples and updated syntax will also be covered.

All too often it seems that the examples presented in conference papers are difficult to translate to "real" code and the reader is forced to bridge the gap between concept and reality. All the examples shown here are copied directly from production code – the only exception being some macro variable references have been "resolved" to show the real values. The hope is that these annotated examples will give some ideas to use in your own jobs and see the breadth of options available to make your output look the way you want it to – without any after-the-fact intervention.

PROC REPORT, ODS PDF and Inline Styles

Many of the examples presented in the paper use PROC REPORT. This procedure has the most flexibility in its use of ODS-related options and can best demonstrate the possibilities of creating PDF output. However, there are similar techniques that can be used with PROC PRINT and PROC TABULATE, as well as a number of techniques we'll see that are procedure independent.

CREATING GROUPS OF DATA COLUMNS

We'll start with something simple – simple that is with ODS and PDF, not so simple just a few short years ago. Notice in Exhibit 1 that there is a little extra space between some of the columns. Not quite a full data column worth, but enough to offset groups of data columns.

This is easy to do in PROC REPORT with some in-line formatting and a computed column. The pertinent code is shown below:

```

column popday Bookings Releases locationgroup,(blank thisyear lastyear diff);
:
define locationgroup / across '' format=securelocation. order=data preloadfmt
                        style=[font_size=9pt font_weight=bold];
define thisyear / analysis sum "&ReportYear" format=comma7.;
define lastyear / analysis sum "%eval(&ReportYear-1)" format=comma7.;
define diff / analysis sum 'Percent^Change' format=MyPct.;
define blank / computed '' format=NoDot. style=[cellwidth=8mm];

compute blank;
  blank = .;
endcomp;

```

Add another column (blank) to those under the location header

The **CELLWIDTH=** STYLE option specifies the width of the data column – in this case 8mm

The **COMPUTE** block creates the variable blank as a missing numeric value on every row of the report

Note in the report that the columns with yearly data (*thisyear*, *lastyear* and *diff*) are placed under a common column header (the “across” variable *locationgroup*). To visually separate the columns we can include a fourth variable, *blank*. *Blank* does not exist in the incoming dataset – it is computed in the procedure and set to missing (note the **COMPUTE** block). This could be done prior to ODS and you’d get a blank column in your output. However, notice the **STYLE=** parameter on the **DEFINE** statement for *blank*: we can specify the width of the column with the **CELLWIDTH** parameter and create a gap as large or small as we need.

We’ll see a number of other examples of style parameters. The syntax is simple:

STYLE=[<style option=style value> <style option=style value >]

You can enclose as many option=value pairs as you need, separated by spaces. A complete list of style options is included in Appendix A at the end of the paper.

CREATING GROUPS OF DATA ROWS

We’ve seen how to add and control the appearance of blank columns in a report. We can use another technique to add blank rows to a report. Again, this technique could be used prior to ODS, but with the addition of **STYLE** we can control the appearance of those blank rows. Take a look at Exhibit 2. Note that the blank line every five rows is a different color than the data rows and that it’s not as tall as the data rows. The blank lines are created in the following **COMPUTE** block.

```

compute before LineGroup / style=[font_size=3pt background=cxBDB76B];
  line ' ';
endcomp;

```

The **FONT_SIZE=** STYLE changes the size of the text that will be generated in the **COMPUTE** block

The **BACKGROUND=** STYLE changes the background color of the text that will be generated in the **COMPUTE** block

The variable *LineGroup* is a non-printing **GROUP** variable that changes values every five observations. So, this **COMPUTE** code executes at the beginning of each group of five observations and writes a blank row to the table. However, it’s size and color are different from the surrounding rows because of the **STYLE=** option.

DIFFERENTIAL STYLES ON DATA ROWS

Another use of inline styles can be seen in Exhibit 3. Item 1 points out that we have different text styles on different rows of the table, depending on the type of data displayed. The total lines are italicized for added emphasis.

The TreatmentDetail variable has a value "XX" for total adult and total youth values. The variable YouthAdult has a value "Z" for the grand total values.

The FONT_STYLE= STYLE changes the appearance of the text for the entire row - note that the scope of the CALL DEFINE is the row (_row_)

```
compute treatmentdetail;
  if YouthAdult eq 'Z' or TreatmentDetail eq 'XX' then
    call define(_row_, 'style', "style=[font_style=italic]");
  if YouthAdult eq 'Z' then call define(_col_, 'style', "style=[background=white]");
endcomp;
```

Note that both of the CALL DEFINES are called for YouthAdult='Z' - you can see in the output that both the subtotal rows and the grand total row are italicized. The subtotal rows do not trigger the second CALL DEFINE and are left with their default background colors.

The row header column normally has a colored background. On the grand total row, we use the BACKGROUND STYLE= option to set the column (_col_) background color to white.

There is one other trick to note from Exhibit 3. We just alluded to the fact that the header row has a background color. It's hard to see in the black and white copy in the paper, but the background color of the header rows are the same color as the bars in the corresponding graph on the bottom of the page. The Adult rows in the table and bars on the Adult graph are green. The background and bars for the youth are blue. We can use another style trick to make this easy.

We start out by defining a SAS format that has RGB color codes as the labels for the data values.

```
value $TxColorDetail
  'OPA', 'IOA', 'MOA', 'GCA', 'XXA' = 'cx52b552'
  'OPY', 'IOY', 'MOY', 'GCY', 'XXY' = 'cx6373b5'
  'OSX' = 'cxd63194'
  'ZZZ' = 'white';
```

All the adult values (ending in "A") are assigned to an RGB value (cx52b552) that is green. The youth values (ending in "Y") are assigned to a value that is blue (cx6373b5).

The format will be used in an unusual place. As we saw above, the dataset used in the report contains a variable called TreatmentDetail that contains the type of treatment received (i.e., OPA=outpatient adult). We're going to reference the format we've just created, which defined the colors we want to use based on the type of treatment received, in the DEFINE statement for TreatmentDetail.

```
column YouthAdult TreatmentDetail (AdmissionDate, Total);

define YouthAdult / group noprint format=$YouthAdult;
define TreatmentDetail / ' ' group preloadfmt order=data
  format=$TreatmentTypeDetail.
  style=[background=$TxColorDetail. foreground=white font_weight=bold];
:
```

Note that the BACKGROUND STYLE= option is calling the format with the colors. The value of TreatmentDetail is going to determine the background color of the cell!

So, using the format in a STYLE= option on the DEFINE statement allows us to control the appearance of the data cell based on the value of the data.

We'll have one more example of inline STYLE= options when we discuss methods for putting output from more than one procedure on a page. For now, let's look at the greatest thing to happen to ODS since.... Well, it's just the greatest thing ever to happen to ODS.

EMBEDDING A HISTOGRAM IN PROC REPORT

Graphics and pictures often help to tell the story that our data contains. There's a little trick that will allow you embed a histogram in PROC REPORT output with nothing more than PROC REPORT code.

The hex character 67 in the Webdings font produces a solid rectangle (■). The key to our little trick is that there is infinitesimal white space between consecutive characters so that when viewed or printed in a PDF document a solid bar is seen.

In the following example, we have a dataset that is pre-aggregated by age group. Our table has a column for age range, count and percent of total. To make the values in the table jump out at the reader, we'll also include a computed column that contains a bar representing the percentage for the age group. The bar will be made up of a string of '67'x characters in the Webdings font.

By trial and error we've determined that the column to hold the bars will be 75mm wide and that we can fit 50 4pt '67'x characters in it. The width of the column is set to 75mm, the font to Webdings and the size to 4pt. Also, make sure that the column is left-justified so that the bars are against the edge of the column and centered vertically so that align visually with the numbers in the table.

```
define Bar / '' style(column)=[cellwidth=75mm font_size=4pt font_face=Webdings  
vjust=middle just=left cellpadding=0];
```

```
compute Bar / char length=200;
  BarSize = round(Percent.sum*50);
  if BarSize gt 0 then
    Bar = repeat('67'x,BarSize);
endcomp;
```

The percent variable (Percent) is pre-calculated, so we can just use Percent.sum to get its value for the current row. We have 50 characters to work with, so the BarSize variable contains the number of characters for each row.

Use the REPEAT function to get the right number of characters for the current bar. Since the REPEAT function always returns at least one character, even if the repeat value is 0, the Bar value is set only if the BarSize is greater than 0. If it is 0, Bar will have a null value and no bar will be displayed. As you can see from the output below, this little trick can give the reader a jump on where to look in the table.

<u>10 Year Age Group (calculated)</u>	<u>Number of Records</u>	<u>Percent</u>
0 - 9	92	22.5%
10 - 19	32	7.8%
20 - 29	34	8.3%
30 - 39	43	10.5%
40 - 49	61	14.9%
50 - 59	64	15.6%
60 - 69	38	9.3%
70 - 79	32	7.8%
80 and older	12	2.9%

There is actually another technique used to display the histogram. In order to get the vertical line to the left of the bars we've turned on a single cell border. This requires setting style attributes in a couple places. On the PROC REPORT statement, we turn off all cell borders with FRAME=VOID and RULES=NONE. Then, in the Percent variable DEFINE statement, we turn on the right-side border with BORDERRIGHTCOLOR=BLACK.

```
proc report ... style(report)=[frame=void rules=none cellpadding=1 cellspacing=0];
:
define Percent / style(column)=[bordertightcolor=black borderrightwidth=1pt];
```

Notice that we had to make the border color assignment in the column preceding the Bar column. The border definitions are applied from right to left, so if we'd set the BORDERLEFTCOLOR attribute for the Bar column, it would have been clobbered by the default right border color in the Percent column. We also had to set the BORDERRIGHTWIDTH attribute, since the CELLSPACING=0 in the PROC REPORT statement takes away the space allocated for borders. A full page of the report can be seen in Exhibit 8.

ODS ESCAPECHAR – THE GREATEST THING SINCE SLICED BREAD

The ODS ESCAPECHAR statement, while not specific to PDF, is one of most powerful new features of the Output Delivery System. The statement defines a character that is used to designate the beginning of a series of formatting commands.

ODS ESCAPECHAR= '~ '

That character will be used to designate sequences of text to be treated as “instructions” for the text that follows them. You can use the escaped sequences to change the style of titles and footnotes, add page numbers and superscripts, highlight single words in your output, and much more. You’ll want to choose a character that is not likely to be used in your data so as not to “confuse” ODS. Common choices are carets (^) and tildes (~). The examples in this paper use a tilde.

We’ll look at a number of examples of using escape sequences to enhance your output. For all the examples we’ll use assume that we’re using the command above and use a caret to denote our escape sequences.

Inline Formatting

We’ve seen a number of examples of formatting with the STYLE= option in PROC REPORT. A major use of ESCAPECHAR is to allow formatting of output almost anywhere – either from procedures, titles and footnotes.

The escape sequence “~{style [...] }” allows you to embed style attributes in the brackets anywhere in your text. Any number of style parameters can be placed inside the brackets. The text between the closing square bracket and the closing curly brace will be formatted using the style attributes listed. (Note: this is a syntax change for inline styles beginning in SAS v9.2. For pre-9.2. syntax, please see the discussion in Lund, 2006).

The following example is from Exhibit 4. Note in the exhibit that the leading part of the text is bold and the number part of the text is not. The code uses the STYLE sequence in the value to be displayed by a LINE statement.

```
compute after key / style={just=left};
  TotalLine = '~{style [font_weight=bold]Total DOC Days for Inmate: }'||put(TotalDays.sum,2.);
  Line ' ';
  Line TotalLine $100.;
endcomp;
```

The variable TotalLine contains the text that will be written out by the LINE statement. We’re embedding text style information using the STYLE escape sequence. In this case, setting the text to bold.

Notice also that there is a STYLE= option on the COMPUTE statement which left justifies the text. You can use both methods for assigning styles. In this case, the ~{style [...]} gives us control over whichever portions of the text need it.

This same exhibit (4), item 1 points to another use of inline styles. The leading line of each section contains a name, booking number, booking date, etc. for inmates in a jail. (The names have been obscured or changed for confidentiality reasons.) Notice that the name is a larger, bold font and that the header text for the other items on the row are bold.

All the information on this row is actually contained in one big text variable. The value of the variable contains all the formatting information, along with the actual data values. Here is the datastep code that creates the variable (Key).

The formatting information is included in the variable itself. All the labels are wrapped with STYLE definitions, while the variable values will be formatted in whatever the default text style is.

The values of the variables (Name, BA, CCN, BookingDate and ReleaseDate), along with the formatting information, are all concatenated together.

```
Key = catt('~{style [font_weight=bold font_size=12pt] ',Name,'}',
          '~{style [font_weight=bold]BA}',BA,
          '~{style [font_weight=bold]CCN}',CCN,
          '~{style [font_weight=bold]Booking Date}',put(BookingDate,mmddyy10.),
          '~{style [font_weight=bold]Release Date}',put(ReleaseDate,mmddyy10.));
```

When this value is printed in PROC REPORT, the formatting information does not print but is applied to the other parts of the value.

This method of inline formatting can be used anywhere text is displayed: titles, footnotes, variable values, etc. It gives you a great deal of control over the appearance of your output.

It should be noted that inline styles are only rendered in text-based output. No SAS/Graph output, from procedures or SAS/Graph statements (i.e., AXIS, LEGEND, TITLE, FOOTNOTE), support them and the text of the style attributes will be printed in the output.

Special Escape Sequences

In addition to {style [...]}, the ESCAPECHAR can also be used to add some special values to your output.

In Exhibit 1 we see a superscript on the footnote at the bottom of the page. Superscripting is achieved with the escape sequence {super nn}, where nn is the value to be superscripted. The code used in Item 3 shows that we can just add the escape sequence to the footnote text.

```
footnote1 '~{super 1}Totals may include contract beds';
```

The 1 is superscripted at the beginning of the text and, again, the escape sequence does not print. As you can guess, there is also a {sub nn} sequence to subscript values.

Exhibit 1 also shows another use of superscripts. Item 1 shows a superscript in a column header. The superscript value is actually contained in the label of the format for these values.

```
value SecureLocation (notsorted)
  1.3 = 'Main Jail - Wall St'
  1.6 = 'Main Jail - Oakes St'
  2   = 'The Ridge'
  9   = 'Subtotal^Secure Beds~{super 1}'
  9.1 = 'Subtotal^Community Corrections'
  9.2 = 'Total~{super 1}';
```

The superscript designations {super 1} can be added to the format labels and will be rendered whenever the formatted values are displayed.

As we've already seen, anywhere text can be generated the escape sequences can be used.

A topic which has generated a number of SUGI/SAS Global Forum papers is putting dynamic page numbers on your output. This is a paper topic no more as it now only an escape sequence added to your titles or footnotes. There are two escape sequences that we'll demonstrate for included page numbers on your output.

A number of the exhibits included in this paper have page numbers – we'll look at those shown on Exhibit 4.

```
footnote h=8pt f=Arial j=1 "%sysfunc(today(),%mddy10.)"
      j=r "~{thispage} / ~{lastpage}";
```

This footnote contains two pieces, the left-justified date and the right-justified page numbers.

The {thispage} sequence resolves to the current page number. The {lastpage} sequence resolves to the page number of the last page, i.e., the number of pages.

We can use the {thispage} and {lastpage} sequences together to get the nice “page x of y” that we’ve struggled so many years for. There is a currently a bug with {lastpage} if there are any graphics on the page. The graphic could be SAS/Graph output that is on the page in a LAYOUT region (we’ll talk about this later) or placed via a PREIMAGE or POSTIMAGE tag in a title or procedure output. If {lastpage} is used, the graphics will not display.

One last set of special escape sequences to note are line breaks and wrapping to a marker. Exhibit 5 shows a nice set of footnotes indented under a header. All of the footnotes in this example are done with a single footnote statement.

We will use two escape sequences here to get the footnotes looking this way. First, the -n sequence produces a line-break in the text. (A quick side note here – does the fact that there are a whole bunch of nicely arranged footnotes generated from a single footnote statement give you a clue that you’ll never have to worry about the 10 footnote limit again!). But, we can use -n in conjunction with another escape sequence (m) to get the indented effect that we see.

The m escape sequence sets a “marker” that the -n will wrap to. This is more easily explained with the code that generated the example. First, notice that this is all one footnote statement. Second, if you look between the code and the exhibit you will see that the individual physical line in the code do not affect the output – they are there only to make them fit on my screen when I’m writing the code. Now, let’s look at the use of the two escape sequences.

```
footnote1 j=1 h=8pt f='Arial'
```

```
'Defi~nitions:'
```

```
'~-nPAO Referral: Referral from police - Source: PAO'
```

```
'~-nCase Credit: Source: OPD'
```

```
'~-nFiling: Referral filed in Superior Court - Source: Superior Court - SCOMIS'
```

```
'~-nResolution: Cases adjudicated in Superior Court by any method (plea, verdict, dismissal or other) '
'- Source: Superior Court - SCOMIS'
```

```
'~-nGuilty Plea: any case that is resolved with a plea irrespective of when in the process the plea '
'occurs (even during trial).'
```

```
'~-nTrial: any case that is resolved with a trial verdict.'
```

```
'~-nDismissal: any case that is dismissed at any time in the process (including those dismissals that '
'occur during trial).'
```

```
'~-nOther: any case resolution that is not resolved via plea or trial - example is a "change in venue".'
```

```
'~-nMedian Days: Median days from filing to resolution. The median is the number of days where half '
'the cases were longer and half the cases were shorter.';
```

Notice the ^m in the middle of the word “Definitions.” This is an escape sequence, which will not print, but marks the place to which any subsequent line breaks should wrap.

The -n characters denote where a line break should take place. Since we’ve set a marker (m) in the text, all the line breaks will wrap to that mark – right under the “n” in “Definitions” in this case. If there were no marker character the -n would still cause a line break, with the text wrapping to the left edge of the page.

There is another line break sequence, {newline n}. This will insert n new lines in the text, the default being one line. There is no {newline} syntax that will break to a marked location – only -n will do that.

SUPERSCRIPTS REVISITED

One drawback of the {super xx} and {sub xx} sequences is that you have no control over the appearance of the text. The size is proportionately adjusted to the size of the preceding text and cannot be changed. None of the other font attributes can be changed. There is an undocumented work around for this – but, as it is undocumented, there is no guarantee that it will continue to work in future releases.

The escape sequences ~nY and nY move the text up and down respectively. Following the move up or down, normal style attributes can be applied. When mimicking a superscript the most common stylistic change will be to decrease the font size. However, there may be times when other changes are desired, such as an italic note with a roman (non-italic) superscript, as shown in the following example:

In this case, we use the SUPER escape sequence to get a superscript. The font style and weight is the same as the rest of the text and the size is based on a pre-determined (and unadjustable) proportion to the original text.

```
title1 f='Helvetica/italic' h=12pt  
"Title Line~{super 123456789} Here";
```

→ Title Line¹²³⁴⁵⁶⁷⁸⁹ Here

Here we take total control over the appearance of the superscript by moving the cursor position with ~nY and changing the attributes of the footnote to be whatever we need it to be.

```
title1 f='Helvetica/italic' h=12pt  
"Title Line~9y~{style [font_size=8pt font_style=roman]}123456789}~9y Here";
```

→ Title Line¹²³⁴⁵⁶⁷⁸⁹ Here

↑
-9y moves the
print position up

↑
With STYLE we change
the font size and style

↑
9y moves the print
position back down
to its original position

ODS TEXT

The ODS TEXT command is like a PUT statement that you can use anywhere. The syntax is simple,

```
ods text='<text goes here>;'
```

The output of the statement will be placed on the page immediately following the last procedure output. It would be a relatively useless command if all it did was place plain text. I bet you've already figured out that all the inline style functionality provided by ESCAPECHAR can be used in PDF TEXT.

Exhibit 6 shows an entire page generated using nothing but PDF TEXT. This offers a wonderful method of adding explanatory text to your reports without having to edit them after the fact.

We start out by turning on bold, 14-point font for the header portion and then turn off the bold for the rest of the text.

```
ods text="~{style [font_weight=bold font_size=14pt]}A. }~{style [font_weight=bold font_size=14pt]}Operation Definition: }  
~{style [font_size=14pt]}The proportion of people in the general population who received publicly funded Outpatient  
mental Health services in the Fiscal Year by RSN."";
```

Note that we've placed a marker here, but there are no line breaks (-n) specified in the text. The text will wrap to the mark even if the line break is caused by an automatic line break due to the length of the text. Note also that the marker cannot be embedded in the style definition, so we need to repeat it for the two pieces of the header.

The ODS TEXT statement can also be made destination-specific. This is done simply by placing the destination name between the other two keywords, i.e. ODS PDF TEXT. This allows for notes that are specific to a destination, if more than one ODS destination is open at a given time.

OUTPUT FROM MORE THAN ONE PROCEDURE ON A PAGE

With SAS/Graph we've had PROC GREPLAY and the VPOS/HPOS/VSIZE/HSIZE options that have allowed putting output from more than one graphic procedure on the page at one time. This has always been more of a challenge with output from text-based reporting procedures. We'll look at two methods now available with ODS which allows us to do just that.

STARTPAGE=NO

By default, any time a new procedure is run a new page will be created in the output document. There is an option (STARTPAGE=) on the ODS PDF statement which controls that page generation.

The STARTPAGE=NO option tells ODS that you don't want a page generated between procedures. You still get a new page when the current page fills up, but new procedure output will start immediately following previous output.

Exhibit 5 shows an example of how this works. The data at the top of the page is generated from a PROC REPORT. The data at the bottom of the page is from a second PROC REPORT. The ODS PDF statement contained a STARTPAGE=NO so that the output from both procedures showed up on the same page.

There is a STARTPAGE=NOW option you can use to force a new page whenever you want it, which is handy if you've turned the startpage off. You can set the startpage to NOW with a ODS PDF statement that does not reference a file, so will be applied to the current PDF file being created. For example,

```
ods pdf file='my pdf file.pdf' startpage=no;
```

```
<procedure 1>
```

```
<procedure 2>
```

```
ods pdf startpage=now;
```

```
<procedure 3>
```

There would be no page break between the output from procedures 1 and 2 above, but output from procedure 3 would begin on a new page.

It was noted earlier that there would be one more STYLE= example when we discussed multiple outputs on a page. In Exhibit 5, the single columns in the upper report need to be centered over the groups of three columns in the lower report. We can do this with STYLE= options on the DEFINE statements in the two procedures.

The important thing to note here is the **CELLWIDTH** values: in the first PROC REPORT, the header column is set to 45 and the “data” columns (Value and Blank) have values of 29 and 17 (46 total).

```
columns Type DateIndex,(Value Blank);
define Type / group '' left style=[cellwidth=45mm font_weight=bold];
define DateIndex / '' across order=internal format=$ReportDateLabels. center;
define Value / '' sum format=CommaUnknown. style=[just=dec cellwidth=29mm] center;
define Blank / '' sum format=Blank. style=[cellwidth=17mm];
```

```
columns ResolutionGroup ResolutionCode DateIndex,(Value Pct MedianDays Blank);
define ResolutionGroup / group noprint;
define ResolutionCode / 'Breakdown of~-2nResolution Dispositions' group style=[cellwidth=45mm];
define DateIndex / '' across order=internal format=$ReportDateLabelsBlank. center;
define Value / "N" sum format=CommaUnknown. right style=[cellwidth=12mm];
define Pct / "%" sum format=PercentUnknown. right style=[cellwidth=13mm];
define MedianDays / "Median~-2nDays" sum style=[cellwidth=13mm];
define Blank / '' sum format=Blank. style=[cellwidth=8mm];
```

We want the columns in the second PROC REPORT to line up with those in the first one. The header column width here is the same (45) and the sum of the “data” columns (Value, Pct, MedianDays and Blank) is the same as above (12+13+13+8=46).

Notice also that we’ve made one more use of -2n. In the variable labels for ResolutionCode and MedianDays, we’ve added a -2n to split the label. Note: in order to fit this all on the page some code has been removed that is part of those labels. This additional information makes the use of the PROC REPORT SPLIT= option unusable in this case.

ODS LAYOUT

Another method of putting more than one piece of output on the page is with ODS LAYOUT. LAYOUT enables you to specify regions on the page that output will be written to. Exhibit 7 shows a report with output from one PROC SQL, three PROC REPORTS and six PROC GCHARTS. How did we do this one?

The first table is generated with PROC SQL with nothing special added, except that STARTPAGE=NEVER is set so that all the other output shows up on the same page. Then we start with LAYOUT. There are three commands necessary to use ODS LAYOUT.

```
ODS LAYOUT START;
```

```
ODS REGION X=xxx Y=yyy WIDTH=www HEIGHT=hhh;
```

```
ODS LAYOUT END;
```

START and END turn on and off the LAYOUT and REGION specifies the size and position of the page region to write output to. There can be as many regions as you want on the page. There are some important region rules to be aware of:

- regions are not transparent. If they overlap, the first one defined has that space on the page and will overwrite the output beneath it.
- the size of the region is determined by the WIDTH and HEIGHT parameters, not the amount of data sent to the region. If you have a large region set and only one row of data, the region is still large (and may overlay other regions).
- output is not sized to the region. If it doesn’t fit, it is truncated. There is a log note warning you of this.
- LAYOUT is limited to one page. There is no spanning of regions or output to additional pages.

With those rules in mind, let’s look at the code for the three PROC REPORTs in Exhibit 7.

```
ods layout start;

ods region width=2.5in height=3in y=.1in x=0in;
    {PROC REPORT code}

ods region width=2.5in height=3in y=.1in x=2.65in;
    {PROC REPORT code}

ods region width=2.5in height=3in y=.1in x=5.5in;
    {PROC REPORT code}

ods layout end;
```

We can have as many regions as we need. In this case, all the regions have the same size (2.5 by 3 in). The X and Y parameters position the output on the page. Notice that all the regions also have the same Y value, so they will be aligned across the top.

Note: the Y value is the distance from the top of the “printable area” to create the region. Since the PROC SQL output has already been sent to the page, the printable area starts below that.

Notice that our regions do not overlap, the X value on each region is greater than the width of the previous region. Also, as noted above, the Y value is set from the top of the page (0=top of printable area). This is opposite of the Y position values in SAS/Graph, where 0 is the bottom of the page.

Beginning in version 9.2, SAS/Graph output is scaled to fit an ODS LAYOUT region. Be aware that though the graphic image itself will scale, if you’ve specified font sizes for text in the graph, they will remain that size. Some trial and error is often necessary to get the font sizes correct.

Even in version 9.3, ODS LAYOUT will still officially be pre-production. The code seems relatively stable and syntax is unlikely to change, but keep that in mind. For more information on using ODS LAYOUT, see Dan O’Connor and Scott Huntley’s 2009 SAS Global Forum paper listed in the references section.

DATA_NULL_REPORTING AND ODS PDF

Traditional DATA_NULL_ reporting, with PUT statements writing the desired output to a file or the output window, has allowed SAS programmers to generate custom reports that can’t be created with a procedure. Now that output can be sent directly to PDF, or any other ODS destination, and can be stylistically enhanced with inline styles. Actually, if you have a data step with a FILE PRINT statement all you need to do is issue an ODS PDF statement before and an ODS PDF CLOSE statement following the data step and the PDF document will be created with the results.

```
ods pdf file='c:\temp\test.pdf';
```

```
data _null_;
    set sashelp.class(where=(sex eq 'M'));
    file print;
```

```
    put Name @20 Height;
run;
```

We think of FILE PRINT as sending results to the output window. Actually, they will go to any open ODS destination. In this case, a PDF file.

```
ods pdf close;
```

Alfred	69
Henry	63.5
James	57.3
Jeffrey	62.5
John	59
Philip	72
Robert	64.8
Ronald	67
Thomas	57.5
William	66.5

All the escape sequences discussed above (style, newline, nspace, etc.) can be added to PUT statements, or in the PUT or to the variable referenced in PUT statements and will be rendered in the PDF output. This makes creating nice looking output quite simple.

Another enhancement to data step reporting is the ODS= option on the FILE PRINT. This options associates the data step output with a table template which allows you to create tabular output. You can actually create output that looks like procedure output using a data step.

Even though it does not look very tabular, Exhibit 9 shows an example of a report created in a data step using a table template. The table in this report has only one column and we'll use a lot of embedded style attributes to format the text the way we want it. This report could have been done with just the simple FILE PRINT and PUT statements method mentioned above except for one thing This method is still restricted by the 256-character LINESIZE limit and the lines in this report are more than 256 characters wide. Using the table template method gets around that limit.

We start by modifying the style template that we'll use. This lets us set some global attributes that we won't have to repeat on each line of text and to set some general appearance attributes for the report.

```
proc template;
  define style work.MyJournal;
  parent=styles.Journal;
  style Data from Cell /
    font = ("SAS Monospace",7pt);
  style Output from Container /
    rules = none
    frame = void;
run;
```

We're going to create our own style template (MyJournal) that has all the attributes of the the parent template (Journal), except for the things we reference in the procedure.

Here we change just a couple things:

- Change the default font for "data cells" (in this report, everything but the titles and footnotes are data cells).
- By default, there would be some borders and lines – turn all those off.

You can learn a lot more about using PROC TEMPLATE to create style templates in Cynthia Zender's 2010 SAS Global Forum paper.

Normally, the FILE PRINT statement sends the results of PUT statements to the output window and/or any open ODS destination.

```
file print ods=(variables=(BigLine
  (label='{~{style [font_size=1pt] }') ));
```

The ODS= option on the FILE PRINT statement lists the variables that will be included in the output "table." In this report, there is only one variable per row. All the data and style information will be part of the variable value.

Note: by default, the variable label is printed above the data column – we'll set the label of the BigLine column to a blank space.

The default table template that is used is called BASE.DATASSTEP.TABLE and most of the time you can use this default. A different template can be specified with the TEMPLATE= parameter of the ODS= option, but you need to understand the way table templates work and the information that they expect. Cynthia Zender's SUGI 30 paper (2005) has a number of great examples of creating tabular output in a dataset using table templates.

In this example, the variable BigLine is the only variable that will be put into the output table that the table template will create. As you can see in Exhibit 9, a lot of information can be put on each line. Here we set up the header for each agency in the report which will be at the top left of each page. All the style information and data values are concatenated together in one big data value.

```
BigLine = '~4P
~{style [font_weight=bold font_size=9pt]Agency: }
~{style [font_size=9pt]'}||
put(Bill_Agency_Cd,$AgencyName.)||
'~{newline 2}';
put _ods_;
```

Note: The 4P inserts a page break in a PDF document.

Remember, in the template above we set the default font to 7pt SAS Monospace. For the first line of each agency, we want a 9pt font, with the header "Agency" in bold. We simply include all the style information in the value of the variable.

The PUT _ODS_ statement writes the value of BigLine to the output destination.

This PUT statement looks different from what we usually see – there is no quoted text or variable names. The _ODS_ specification is a reference back to the FILE PRINT statement. All the variables listed in the VARIABLES parameter are written to the output file. In this example, there is only one variable, BigLine, listed there and its value will be written to the PDF document.

```
4P~{style [font_weight=bold font_size=9pt]Agency: }~{style [font_size=9pt]Pixley }~{newline 2}
```

The value of BigLine is shown above, but in the PDF it is rendered with the desired format attributes.

→ **Agency: Pixley Police Department**

Following the agency header on each page are rows of information about each person booked into the jail by that agency. The first row for each person has the booking number, name and dates of incarceration. We want that row to have a gray background and, like the agency row, have header text in bold and the data values in normal text. Again, we put all the necessary information into the value of BigLine and use PUT _ODS_ to write it to the PDF document.

```
BigLine = '~{style [font_weight=bold background=cxf0f0f0]}'||
put (BookingNum,10.)||' '||
put (InmateName,$45.)||
'~{style [font_weight=bold]Book-Release: }'||
BookRelease||
' '||Res_Status;
put _ods_;
```

Just like we did above, the style information and data are combined in the variable value and the PUT _ODS_ statement writes it to the PDF.

Note: the background color set in the first STYLE segment affects the whole cell. Since we have a single cell row, the entire row has the gray background.

We similarly create a value of BigLine that has the arrest and charge information, with embedded styles, that will go on the next line(s). The results are quite nice. Notice in the partial output below that all the style instructions

Agency: Pixley Police Department

2010009277	ADAMSON, JOHN	Book-Release: 05/12/2010:19:14
Arr/Chg: 1/1	RESISTING ARREST	Court/Case: TL/
Arr/Chg: 2/1	CRIMINAL MISCHIEF [LESS THAN \$1,000]	Court/Case: TL/

have been rendered: font sizes are changed, the bold text is turned “on” and “off” and that the gray background is added to the first row of each person’s information. All of this is achieved by embedding the style information in the variable values themselves.

THE FUTURE OF DATA_NULL_ REPORTING – THE REPORT WRITING INTERFACE

It’s pretty cool to be able to create formatted, tabular reports from a data step. But, in the new world of DATA_NULL_ reporting you can define tables right in the data step code and have many different table on the same page of output. Before we begin a discussion of the Report Writing Interface (RWI), please understand that this section is just to get your interest piqued. This is huge topic and you can get much more information in Dan O’Connor’s 2009 SAS Global Forum paper listed in the references section. It contains 40 pages on this topic alone.

The RWI uses a data step object called ODSOut. There are “methods” (like functions) of that object that will create tables, rows, cells, text, page breaks, lines, etc. To use an ODSOut object it is first declared and given a name – this only has to be done once in the data step and is routinely placed in a conditional section of code:

```
if _n_ eq 1 then
do;
declare odsout obj(); ←
<other statements>
end;
```

This gives us an ODSOut object named “obj” – we’ll use that name to reference methods that build our output.

Again, the DECLARE statement only have to be executed once in the data step.

Once the object is declared you can call methods that perform different tasks. For instance, with our object “obj,” just a few of the possible methods are:

obj.table_start()	- begins a table (there is a table_end method that closes a table)
obj.row_start()	- begins a row in that table – you can have as many rows in the table as you want (there is also a row_end method that closes a row)
obj.format_cell()	- inserts a cell (column) into that row – you can have as many cells in a row as you

	want, but each row must have the same number of cells
obj.format_text()	- inserts a line of text (not part of a table)
obj.line()	- puts a horizontal line on the page
obj.page()	- inserts a page break

Note that all of these methods have parameters that can be placed in the parentheses. For example, the Format_Cell method has a “text” parameter that specifies the text to be printed in the cell. You can specify style attributes in most method calls as well, specifying cell borders, appearance of text, line widths, etc., with the OVERRIDES parameter. The report shown in Exhibit 10 uses the RWI and each page can have as many as 20 or more separate tables.

The report contains information about inmates housed in a county jail that have used extra-fee services (infirmary, psych unit, guarding at a hospital, etc.). It reports, by inmate, the charges that they have and the details about the premium fee services. Each type of information presented has data elements and this makes using a simple tabular output difficult. The RWI is a perfect tool for this report because we can use a different table structure for each type of information. The segment of the report shown below contains six separate types of output : a line, a text string and four tables.

Coffelt, Marie S				Inmate #: 0244970	← A horizontal line
Charge Information: Booking #: 210021612		#1 ASSAULT #2 FTA/DWLS	Direct (MD) Warrant (MB)	06/26/2010 - 06/28/2010 06/26/2010 - 06/28/2010	Tables with
Premium Locations: Booking #: 210021612		Infirmary	06/26/2010 - 06/28/2010		• Inmate information (1 row)
1:1 Guarding: Booking #: 210021612		06/26/2010	Guards: 1	Hours: 8	• Charge information (1 to many rows)
Billing Total: \$1,242.37		Maintenance (days): \$320.10 (3)	Infirmary (days): \$482.67 (3)	1:1 Guarding (hours): \$439.60 (8)	• Premium location information (0 to many rows)
					• Guarding information (0 to many rows)
					Formatted text string with final tally information

So, how did we get those results? Let's look at each of those types of output here. There is a horizontal line between the set of data for each inmate.

```
if first.InmateNum then
do;
<more code>

obj.line();

<more code>
```

Getting the horizontal line is simple – the LINE() method draws a horizontal line across the page. It's placed in a block of code that runs on the first record of an inmates data.

There is an optional parameter, SIZE, for the LINE() method that controls the thickness of the line. The color and line pattern cannot be changed at this time.

The inmate information is put into a table, even though there is always only one row. The main reason for this is to make the inmate numbers have the same horizontal justification from inmate to inmate. The report uses a proportional font, so we can't just pad the name with blanks before putting the inmate number. So, a single-row, two-cell table is used.

```
obj.table_start(overrides: 'frame=void rules=None' );
obj.row_start();
obj.format_cell(text: InmateName);
obj.format_cell(text: catx(' ', 'Inmate #:', InmateNum));
obj.row_end();
obj.table_end();
```

The creation of a table is pretty straightforward:

- Start the table (turning off all the cell borders)
- Start the first, and only, row
- Insert a cell with the inmates name
- Insert a cell with the inmate number
- End the row
- End the table

The inmate information, like the LINE() method call, is also in the block of text that runs on the first record for each inmate. Note that there are additional attributes, set in the OVERRIDE parameter, on the TABLE_START and FORMAT_CELL calls that are not shown because of space here. They control things like the width of the cell, the justification of the text, the font size and style, etc.

The charge data is also in a table, with a row for each charge that the inmate has. The components of the table are split up a little bit here. The TABLE_START is called inside the first.InmateNum block, along with the line and inmate information. The table rows and cells are built on each iteration of the data step and the TABLE_END is called in a block of code for last.InmateNum.

```

if first.InmateNum then
do;
  <more code>

  obj.table_start(overrides: 'frame=void rules=none' );
  obj.format_text(text: '~{newline}Charge Information:');
end;

obj.row_start();
if BookingNum ne PreviousBN then BN_Info = catx(' ','Booking #:',BookingNum);
else BN_Info = '';
obj.format_cell(text: BN_Info);
obj.format_cell(text: ChargeInfo);
obj.format_cell(text: TOW);
obj.format_cell(text: ChargeDateInfo);
obj.row_end();

if last.InmateNum then
do;
  <more code>
  obj.table_end();
  <more code>
end;

```

Along with the inmate information table, the start of the table of charges is placed inside the first.InmateNum block of code.

Immediately following the start of the table, before we've started the first row, we place a line of text, using the FORMAT_TEXT method, that has the header for the section. We use the ~{newline} escape sequence to put a blank line between the inmate information table and the start of the charge information.

For every observation for the inmate, we create a row in the table. The row will have four cells (columns):

1. Booking number – notice that this is conditional and will only print on the first record of each booking
2. The charge information contains the charge number and description
3. The type of charge, including whether it a misdemeanor or felony
4. The date range that the charges were in effect

Finally, in a block of code for last.InmateNum, we close the table of charge information.

The tables for the premium location and guarding data are built in much the same way. If there is no data for one or the other of those tables, nothing is printed.

Finally, also inside the block of code that runs for last.InmateNum, a line of text is written containing the summary information. This is done with a FORMAT_TEXT method call similar to the way the “Charge Information” header was done in the example above.

This is a very quick run-through of an RWI report. We have up to four tables per inmate, each with different number of rows and columns. It's not the simplest example we could have run through, but it shows the power of this technique. As with ODS LAYOUT, the Report Writing Interface is still pre-production and will be even in version 9.3. However, this example shows it can still do quite a bit right now. Please take a look at Dan's paper for a lot of good examples and a more thorough coverage of the methods and options available.

CONCLUSION

This paper has presented a number of tips and tricks that you can use to enhance to look of your SAS output. There are a number of other options available for ODS in general and PDF output in particular. I'd encourage you to go to the SAS website where there is a wealth of information. Go to support.sas.com and click on Communities and Base SAS. There you'll find FAQs, white papers, news and other information on ODS. Past SUGI/SAS Global Forum papers are a wonderful resource for most any subject related to SAS. ODS-related topics are no exception. Just a few that offer additional information on the topics covered in this paper are listed in the following reference section.

REFERENCES

Lund, Pete, “PDF Can Be Pretty Darn Fancy: Tips and Tricks for the ODS PDF Destination,” *Proceedings of the Thirty-First Annual SAS Users Group International Conference*, SAS Institute Inc. (Cary, NC), 2006.
<http://www2.sas.com/proceedings/sugi31/092-31.pdf>

O'Connor, Daniel, “The Power to Show: Ad Hoc Reporting, Custom Invoices, and Form Letters,” *Proceedings of the 2009 SAS Global Forum Conference*, SAS Institute Inc. (Cary, NC), 2009.

(http://support.sas.com/rnd/base/datastep/dsubject/Power_to_show_paper.pdf - this is an updated version of the paper presented at the conference)

O'Connor, Daniel and Huntley, Scott, "Breaking New Ground with SAS® 9.2 ODS Layout Enhancements," *Proceedings of the 2009 SAS Global Forum Conference*, SAS Institute Inc. (Cary, NC), 2009.
(<http://support.sas.com/resources/papers/proceedings09/043-2009.pdf>)

Kevin D. Smith, "PROC TEMPLATE Tables from Scratch," *Proceedings of the 2007 SAS Global Forum Conference*, SAS Institute Inc. (Cary, NC), 2007.
(<http://www2.sas.com/proceedings/forum2007/221-2007.pdf>)

Zender, Cynthia, "The Power of TABLE Templates and DATA _NULL_," *Proceedings of the Thirtieth Annual SAS Users Group International Conference*, SAS Institute Inc. (Cary, NC), 2005.
(<http://www2.sas.com/proceedings/sugi30/088-30.pdf>)

Zender, Cynthia, "SAS® Style Templates: Always in Fashion," *Proceedings of the 2010 SAS Global Forum Conference*, SAS Institute Inc. (Cary, NC), 2010.
(<http://support.sas.com/resources/papers/proceedings10/033-2010.pdf>)

AUTHOR CONTACT INFORMATION

Pete Lund
Looking Glass Analytics
215 Legion Way SW
Olympia, WA 98501
(360) 528-8970
pete.lund@lgan.com

ACKNOWLEDGEMENTS

SAS® is a registered trademark of SAS Institute, Inc. in the USA and other countries. Other products are registered trademarks or trademarks of their respective companies.

1 – Superscript in column headers

Exhibit 1

2 – Gaps between data columns

Snohomish County Department of Corrections Daily Secure Population September 2005

	Main Jail - Wall St			Main Jail - Oakes St			The Ridge			Subtotal Secure Beds ¹			Subtotal Community Corrections			Total ¹		
	September	Bookings	Releases	2005	2004	Percent Change	2005	2004	Percent Change	2005	2004	Percent Change	2005	2004	Percent Change	2005	2004	Percent Change
1	82	69		448	532	-15.7%	590		156	1,038	888	50.9%	75	54	38.5%	1,113	742	50.0%
2	73	67		458	542	-15.4%	600		147	1,058	889	53.5%	74	58	27.5%	1,132	747	51.5%
3	65	47		430	544	-20.9%	626		152	1,058	888	51.7%	76	58	31.0%	1,132	754	50.1%
4	38	57		475	538	-11.7%	591		150	1,068	888	54.9%	78	60	30.0%	1,144	748	52.9%
5	51	44		469	554	-15.3%	585		150	1,054	704	49.7%	78	58	34.4%	1,132	762	48.5%
6	75	87		477	569	-16.1%	593		144	1,070	713	50.0%	73	55	32.7%	1,143	768	48.8%
7	83	91		454	583	-23.4%	601		139	1,055	732	44.1%	76	53	43.3%	1,131	785	44.0%
8	73	81		468	559	-16.2%	578		141	1,048	700	48.4%	77	50	54.0%	1,123	750	48.7%
9	68	72		447	562	-20.4%	591		135	1,038	897	48.9%	77	51	50.9%	1,115	748	48.0%
10	44	44		451	545	-17.2%	578		151	1,029	886	47.8%	76	57	33.3%	1,105	753	46.7%
11	38	45		444	565	-21.4%	559		149	1,033	714	44.8%	75	62	20.9%	1,108	778	41.3%
12	101	70		448	567	-20.9%	576		153	1,024	720	42.2%	76	58	31.0%	1,100	778	41.3%
13	67	78		468	563	-16.8%	585		151	1,053	714	47.4%	76	59	28.8%	1,129	773	46.0%
14	85	87		448	557	-18.5%	592		150	1,040	707	47.1%	75	59	27.1%	1,115	768	45.5%
15	72	93		425	535	-20.5%	583		146	1,008	880	48.2%	100	83	58.7%	1,108	743	48.1%
16	80	64		433	539	-19.8%	588		134	1,021	873	51.7%	76	93	-18.2%	1,097	768	43.2%
17	47	36		440	522	-15.7%	604		144	1,044	866	56.7%	75	59	27.1%	1,119	725	54.3%
18	38	38		438	528	-17.0%	612		148	1,050	878	55.3%	75	61	22.9%	1,125	737	52.6%
19	77	110		455	531	-14.3%	590		153	1,045	884	52.7%	74	81	21.3%	1,119	745	50.2%
20	65	78		457	543	-15.8%	588		148	1,025	891	48.3%	72	59	22.0%	1,097	750	46.2%
21	108	89		435	521	-16.5%	591		143	996	864	50.0%	73	59	23.7%	1,069	723	47.8%
22	67	100		426	517	-17.6%	567		142	993	859	50.8%	102	82	64.5%	1,085	721	51.8%
23	81	89		428	535	-20.0%	588		141	988	878	45.8%	72	108	-32.0%	1,058	782	35.2%
24	35	35		442	530	-16.6%	587		145	1,009	875	49.4%	72	88	9.0%	1,081	741	45.8%
25	44	32		443	524	-15.4%	594		144	1,007	868	50.7%	71	64	10.9%	1,078	732	47.2%
26	78	81		438	541	-18.0%	578		146	1,018	888	48.1%	69	61	13.1%	1,085	747	45.2%
27	92	73		441	558	-20.9%	576		140	1,017	868	45.7%	70	59	18.6%	1,087	757	43.5%
28	80	75		420	555	-24.3%	585		131	1,005	888	48.5%	92	59	55.9%	1,097	745	47.2%
29	72	108		455	554	-17.8%	591		132	1,016	888	48.1%	83	94	-1.0%	1,109	780	42.1%
30	77	51		427	547	-21.8%	570		130	997	877	47.2%	72	91	-20.8%	1,069	768	39.1%
MTD	68.3	69.0		446.3	545.7	-18.2%	583.6		144.4	1,029.8	880.1	49.2%	77.3	63.8	21.5%	1,107.3	753.7	48.8%
QTD	68.0	65.6		446.9	555.3	-20.9%	550.4		151.4	997.3	716.7	39.1%	75.8	61.1	23.9%	1,073.1	777.8	37.9%
YTD	60.7	59.5		543.7	594.9	-8.6%	228.4		81.3	851.4	773.0	10.1%	71.7	60.5	18.4%	823.1	833.5	18.7%

¹ Totals may include contract beds

3 – Superscript in footnote text

Exhibit 2

Arrestee Drug Abuse Monitoring (ADAM) Report Drug Test Results, By Drug By Site Adult Male Arrestees, 2000 through 2003 - Revised Weights

Primary City	Percent of Arrestees Who Tested Positive for:						
	Any NIDA-5 Drug ¹	Marijuana	Cocaine Crack	Opiates Heroin	PCP	Methamphetamine	Multiple NIDA-5 Drugs
Albany, NY	67.3%	49.4%	29.3%	5.3%	0.5%	0.0%	16.6%
Albuquerque, NM	64.1%	39.1%	36.5%	12.4%	0.4%	8.1%	26.4%
Anchorage, AK	57.0%	42.8%	21.6%	4.4%	0.0%	0.8%	11.5%
Atlanta, GA	70.1%	38.2%	48.0%	2.9%	0.0%	1.7%	19.7%
Birmingham, AL	64.4%	43.5%	33.7%	7.9%	0.0%	0.6%	19.3%
Boston, MA	76.2%	44.5%	34.8%	17.8%	0.0%	0.0%	18.0%
Charlotte, NC	63.3%	44.3%	34.3%	2.9%	0.0%	0.4%	17.5%
Chicago, IL	83.0%	49.0%	46.2%	25.7%	2.5%	0.5%	34.7%
Cleveland, OH	70.5%	48.2%	35.9%	4.4%	7.7%	0.5%	23.8%
Dallas, TX	57.9%	36.2%	30.6%	6.0%	3.3%	4.1%	18.6%
Denver, CO	63.1%	40.6%	34.9%	4.7%	0.6%	3.7%	19.4%
Des Moines, IA	58.2%	43.3%	10.3%	2.4%	1.3%	21.3%	19.0%
Detroit, MI	65.9%	47.6%	23.5%	7.7%	0.0%	0.0%	11.6%
Ft. Lauderdale, FL	61.6%	43.6%	30.5%	2.2%	0.0%	0.0%	14.6%
Honolulu, HI	62.0%	31.1%	12.3%	4.6%	0.1%	39.3%	22.2%
Houston, TX	56.3%	35.6%	30.0%	6.5%	3.9%	0.9%	16.6%
Indianapolis, IN	65.0%	47.8%	33.0%	4.4%	1.3%	1.0%	20.8%
Kansas City, MO	69.2%	49.5%	33.4%	0.3%	8.7%	1.0%	22.7%
Laredo, TX	52.1%	26.4%	39.3%	9.8%	0.0%	0.0%	20.2%
Las Vegas, NV	62.0%	34.1%	22.8%	5.4%	2.1%	22.4%	20.8%
Los Angeles, CA	66.4%	39.0%	26.0%	3.0%	1.0%	24.8%	25.3%
Manhattan, NY	75.5%	41.9%	43.8%	16.9%	1.7%	0.2%	25.4%
Miami, FL	62.7%	38.6%	44.9%	3.4%	0.0%	0.1%	23.7%
Minneapolis, MN	68.2%	51.9%	29.0%	4.8%	2.2%	2.8%	20.5%
New Orleans, LA	70.6%	46.0%	40.3%	15.3%	0.1%	1.0%	25.8%
Oklahoma City, OK	70.4%	63.8%	24.1%	3.7%	4.1%	12.1%	24.3%
Omaha, NE	66.1%	47.9%	20.9%	3.2%	0.5%	17.6%	21.5%
Philadelphia, PA	70.5%	44.8%	34.7%	13.0%	8.7%	0.2%	26.1%
Phoenix, AZ	69.9%	38.6%	27.6%	5.5%	1.3%	26.4%	26.9%
Portland, OR	66.7%	36.2%	25.0%	12.8%	0.7%	21.9%	24.8%
Rio Arriba, NM	71.7%	44.4%	35.4%	26.9%	0.0%	1.9%	30.0%
Sacramento, CA	75.1%	49.2%	18.8%	6.2%	1.2%	31.7%	28.4%
Salt Lake City, UT	55.1%	33.0%	17.2%	6.7%	0.0%	20.2%	19.0%
San Antonio, TX	55.6%	39.1%	28.0%	9.2%	0.0%	2.3%	20.0%
San Diego, CA	62.2%	38.3%	12.0%	5.8%	0.8%	29.2%	21.4%
San Jose, CA	58.2%	35.6%	12.8%	3.5%	2.8%	26.3%	21.3%
Seattle, WA	65.9%	36.0%	35.0%	9.4%	2.2%	10.6%	23.0%
Spokane, WA	63.1%	43.0%	16.0%	8.0%	1.0%	23.7%	23.5%
Tampa, FL	59.6%	45.2%	29.7%	3.7%	0.2%	1.5%	19.4%
Tucson, AZ	68.6%	44.6%	40.2%	6.3%	0.6%	8.6%	27.9%
Tulsa, OK	69.0%	51.8%	20.6%	4.8%	2.1%	16.1%	23.2%
Washington, DC	64.2%	37.6%	27.2%	9.9%	10.4%	0.4%	19.0%
Woodbury, IA	40.8%	29.8%	7.3%	0.7%	0.0%	14.4%	10.8%
Median	65.0%	43.0%	29.7%	5.5%	0.7%	2.3%	21.3%

1 – Groups of rows

2 – gap color and size

* Results are suppressed if fewer than 25 arrestees were tested for a particular drug.

¹ The five drugs listed here are referred to as the NIDA-5, established by the National Institute on Drug Abuse as a standard panel of commonly used illegal drugs.

Exhibit 3

1 – Differential text style



2 – Text background matches graph bars

1 – Style info
embedded in data

Exhibit 4

WA State Department of Corrections Contract Violators
DAJD Detail Billing Report

September 2005

BA: [REDACTED] CCN: [REDACTED] Booking Date: 09/08/2005 Release Date: 09/13/2005									
From-To	Days	Reason	Chg Description	Status	RCW	TOW	ORI	Cause	Case
9/8-9/13	6	OAA only	1 COMM. PLACEMENT ARR	PHS	9.94A.195	FS	WA034015G		865718
Total DOC Days for Inmate: 6									
BA: [REDACTED] CCN: [REDACTED] Booking Date: 08/24/2005 Release Date: 09/13/2005									
From-To	Days	Reason	Chg Description	Status	RCW	TOW	ORI	Cause	Case
9/1-9/13	13	OAA only	1 COMM. PLACEMENT ARR	PHS	9.94A.195	FS	WA034015G		869732
Total DOC Days for Inmate: 13									
BA: [REDACTED] CCN: [REDACTED] Booking Date: 09/28/2005 Release Date:									
From-To	Days	Reason	Chg Description	Status	RCW	TOW	ORI	Cause	Case
9/28-9/30	3	OAA only	1 COM PLMT ARR*ASSLT2	PHS	9.94A.195	FS	WA034015G		887187
Total DOC Days for Inmate: 3									
BA: [REDACTED] CCN: [REDACTED] Booking Date: 08/24/2005 Release Date: 09/22/2005									
From-To	Days	Reason	Chg Description	Status	RCW	TOW	ORI	Cause	Case
9/1-9/7	7	OAA only	1 ESCAPE/VCSA	PHS	9.94A.195	FS	WA034015G		851456
9/8-9/22	15	OAA only	2 COMM. PLACEMENT REVO	SJS	9.94A.270	FS	WACORR000	041123083	851456
Total DOC Days for Inmate: 22									
BA: [REDACTED] CCN: [REDACTED] Booking Date: 09/03/2005 Release Date: 09/22/2005									
From-To	Days	Reason	Chg Description	Status	RCW	TOW	ORI	Cause	Case
9/8-9/18	10	OAA only	1 INVEST THEFT	INV	2389	IB	WASPD0000		050377683
9/19-9/22	4	OAA only	2 COMM. PLACEMENT ARR	PHS	9.94A.195	FS	WA034015G		981581
			3 COMM. PLACEMENT REVO	PHR	9.94A.270	FS	WA034015G		981581
Total DOC Days for Inmate: 14									

2 – Embedded
style info in
LINE statement

3 – Page
numbers

1 – Multiple sets of
columns aligned

Exhibit 5

King County Felony Case Processing Summary Report Based on Resolutions through August 2005

	2005			YTD through August 2004			YTD through August 2003			2004			2003		
	YTD	N	%	Median Days	N	%	Median Days	N	%	Median Days	N	%	Total	N	%
PAO Referrals	8,610			9,578			9,815			14,308			14,462		
Case Credits Assigned	6,291			6,077			5,836			8,877			8,813		
Filings	6,557			6,778			6,655			9,974			10,326		
Resolutions	6,144			6,900			5,975			9,830			8,944		
Breakdown of Resolution Dispositions															
Guilty Plea	4,825	78.5%		70.0	5,420	78.6%	63.0	4,925	82.4%	52.0	7,751	78.9%	63.0	7,335	82.0%
Prior to Trial Setting	3,549	57.9%		53.0	3,490	50.6%	42.0	3,195	53.5%	35.0	5,090	51.8%	43.0	4,795	53.6%
After 1st trial date set	1,276	20.9%		135.0	1,930	28.0%	120.0	1,730	29.0%	110.5	2,661	27.1%	124.0	2,540	28.4%
Dismissal	971	15.8%		106.0	1,111	16.1%	82.0	729	12.2%	77.0	1,545	15.7%	83.0	1,143	12.8%
Prior to Trial Setting	650	10.6%		86.0	650	9.4%	49.5	422	7.1%	42.0	947	9.6%	52.0	673	7.5%
After 1st trial date set	321	5.2%		130.0	461	6.7%	132.0	307	5.1%	118.0	598	6.1%	132.0	470	5.3%
Trial	336	5.5%		230.5	353	5.1%	201.0	313	5.2%	176.0	512	5.2%	201.5	445	5.0%
Other	12	0.2%		123.0	16	0.2%	0.0	8	0.1%	186.0	22	0.2%	0.0	21	0.2%

Definitions:

PAO Referral: Referral from police - Source: PAO

Case Credit: Source: OPD

Filing: Referral filed in Superior Court - Source: Superior Court - SCOMIS

Resolution: Cases adjudicated in Superior Court by any method (plea, verdict, dismissal or other) - Source: Superior Court - SCOMIS

Guilty Plea: any case that is resolved with a plea irrespective of when in the process the plea occurs (even during trial).

Trial: any case that is resolved with a trial verdict.

Dismissal: any case that is dismissed at any time in the process (including those dismissals that occur during trial).

Other: any case resolution that is not resolved via plea or trial - example is a "change in venue".

Median Days: Median days from filing to resolution. The median is the number of days where half the cases were longer and half the cases were shorter.

Exhibit 6

Access I.A. Community Outpatient Penetration Rates - General Population

A. Operation Definition: The proportion of people in the general population who received publicly funded Outpatient mental health services in the Fiscal Year by RSN.

Operational Measure: This is calculated by dividing the number people who recieved the outpatient mental health services during a Fiscal Year by an estimate of the general population as of April 1st of the calendar year.

Formulas:

Number of person receiving the outpatient mental health services during the Fiscal Year by RSN

Estimate of people in the general population in that Fiscal Year by RSN

Data Notes:

2002 and 2003 data excludes crisis hotline calls, 24-hour crisis services , and residential services as specified in the January 2002 Data Dictionary. Reporting of these services varies across the state.

The State total is unduplicated clients across all RSNs (i.e., each person is only counted once in the State).

The RSN count shows the number of unduplicated clients within each of RSN (i.e., one person is counted in each RSN in which they received services.).

Counts are of people, not admissions, episodes, or units of service.

Population are from the Washington State Office of Financial Management (OFM). The calendar year of estimate used is the same as the analysis year. For example, for Fiscal Year 2004, Calendar year 2004 estimates would be used.

OFM releases complete population estimates every alternate year. The intervening years were estimated by assuming a constant rate of change between those years.

Exhibit 7

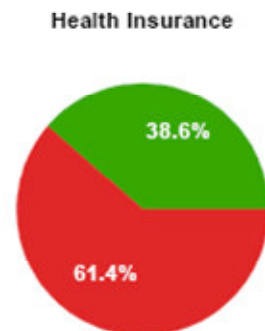
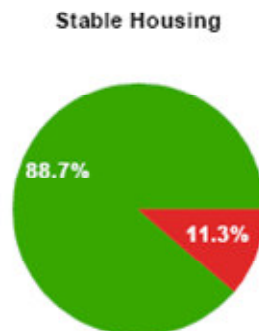
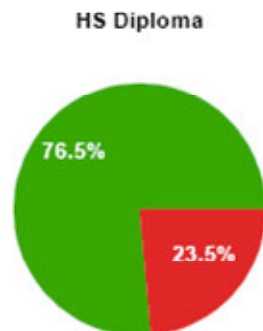
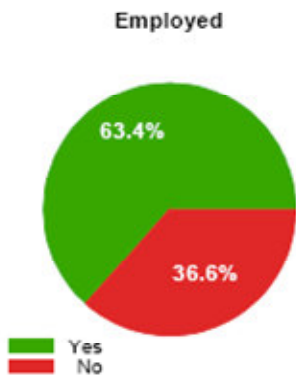
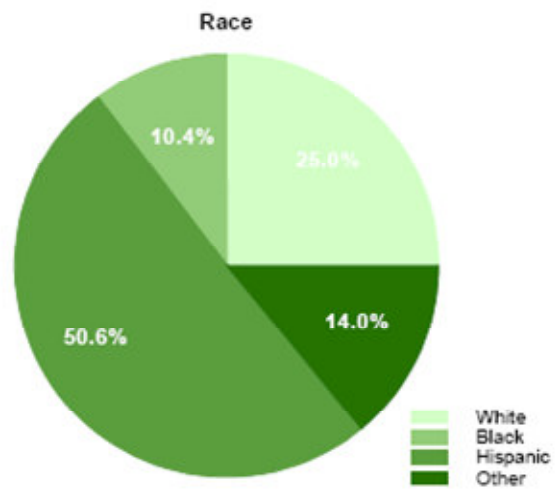
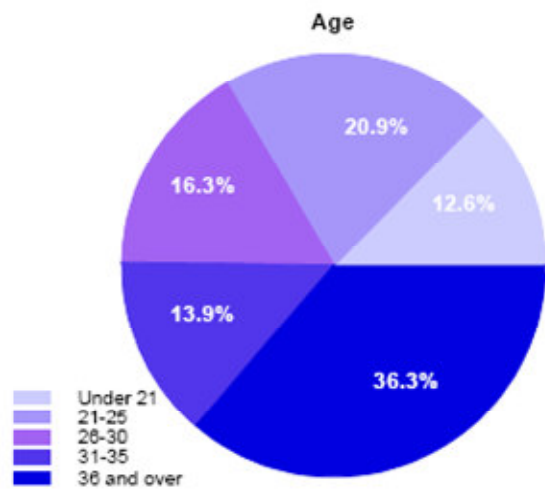
Arrestee Drug Abuse Monitoring (ADAM) Report Demographic and Sociodemographic Characteristics

ADAM Site: San Jose, CA
Interview Year: 2000 through 2003
Sample: Adult Male
Sample Size (unweighted): 3,164
Breakout Category: None
Analysis Weights: Revised Weights

Age	
Under 21	12.6%
21-25	20.9%
26-30	16.3%
31-35	13.9%
36 and over	36.3%

Race	
White	25.0%
Black	10.4%
Hispanic	50.6%
Other	14.0%



















Other Characteristics	
Unemployed	36.6%
No High School Diploma	23.5%
Unstable Housing	11.3%
No Health Insurance	61.4%












10/05/2005

Surveillance Demographics Report
For the Month-Year Reported*
Beginning 01/2005 and Ending 12/2010

Reporting Area: Guam
Site Name: All Sites

<u>5 Year Age Group (calculated)</u>	<u>Number of Records</u>	<u>Percent</u>	
0 - 4	62	15.2%	
5 - 9	30	7.3%	
10 - 14	13	3.2%	
15 - 19	19	4.6%	
20 - 24	12	2.9%	
25 - 29	22	5.4%	
30 - 34	22	5.4%	
35 - 39	21	5.1%	
40 - 44	27	6.6%	
45 - 49	34	8.3%	
50 - 54	26	6.4%	
55 - 59	38	9.3%	
60 - 64	24	5.9%	
65 - 69	14	3.4%	
70 - 74	17	4.2%	
75 - 79	15	3.7%	
80 - 84	5	1.2%	
85 and older	7	1.7%	
.	1	0.2%	

<u>10 Year Age Group (calculated)</u>	<u>Number of Records</u>	<u>Percent</u>	
0 - 9	92	22.5%	
10 - 19	32	7.8%	
20 - 29	34	8.3%	
30 - 39	43	10.5%	
40 - 49	61	14.9%	
50 - 59	64	15.6%	
60 - 69	38	9.3%	
70 - 79	32	7.8%	
80 and older	12	2.9%	
.	1	0.2%	

* Report includes verified cases only

Exhibit 9

Mayberry County Jail Mt Pilot Inmate Premium Location Detail Report

June 2010

Bednar, Darren Q Inmate #: 0045258

Charge Information:
Booking #: 210019728 #1 FTA/THEFT Warrant (MB) 06/10/2010 - 06/11/2010

Premium Locations:
Booking #: 210019728 Other Psych 06/10/2010 - 06/11/2010

Billing Total: \$345.20 Maintenance (days): \$213.40 (2) Other Psych (days): \$131.80 (2)

Belue, Javier L Inmate #: 0844429

Charge Information:
Booking #: 210019859 #1 EXCLUSION Direct (MD) 06/11/2010 (still open)

Premium Locations:
Booking #: 210019859 Other Psych 06/11/2010 - 06/30/2010

Billing Total: \$3,452.00 Maintenance (days): \$2,134.00 (20) Other Psych (days): \$1,318.00 (20)

Bowie, Carlos M Inmate #: 0012581

Charge Information:
Booking #: 210019794 #1 CRIM TRESPASS Direct (MD) 06/11/2010 - 06/15/2010

Premium Locations:
Booking #: 210019794 Psych Unit 06/11/2010 - 06/14/2010
Other Psych 06/15/2010 - 06/15/2010

Billing Total: \$1,481.56 Maintenance (days): \$533.50 (5) Psych Unit (days): \$882.16 (4) Other Psych (days): \$65.90 (1)

Bratcher, Donald K Inmate #: 0474276

Charge Information:
Booking #: 210020905 #1 DWLS 3 BNOTE Direct (MD) 06/20/2010 - 06/22/2010
#2 TRIP PERMIT VIO Direct (MD) 06/20/2010 - 06/22/2010

Premium Locations:
Booking #: 210020905 Other Psych 06/20/2010 - 06/21/2010

Billing Total: \$345.20 Maintenance (days): \$213.40 (2) Other Psych (days): \$131.80 (2)

Busbee, Lonnie R Inmate #: 0129242

Charge Information:
Booking #: 210018888 #1 THEFT Direct (MD) 06/03/2010 - 06/18/2010
#2 CRIM TRESPASS Direct (MD) 06/03/2010 - 06/18/2010

Premium Locations:
Booking #: 210018888 Other Psych 06/03/2010 - 06/18/2010

Billing Total: \$2,654.90 Maintenance (days): \$1,000.50 (15) Other Psych (days): \$1,054.40 (10)

Coffelt, Marie S Inmate #: 0244970

Charge Information:
Booking #: 210021612 #1 ASSAULT Direct (MD) 06/26/2010 - 06/28/2010
#2 FTA/DWLS Warrant (MB) 06/26/2010 - 06/28/2010

Premium Locations:
Booking #: 210021612 Infirmary 06/26/2010 - 06/28/2010

1:1 Guarding:
Booking #: 210021612 06/26/2010 Guards: 1 Hours: 8

Billing Total: \$1,242.37 Maintenance (days): \$320.10 (3) Infirmary (days): \$482.07 (3) 1:1 Guarding (hours): \$439.00 (8)

Notes:
• Charges shown are all Mt Pilot charges open during June.
• Premium location days and guarding hours shown are only those that occurred in June.

**County Sheriff's Office
Corrections Bureau
Daily Billing Report**

Agency: Hooterville Police Department

2010010677	ACREE, BRUCE	Book-Release: 06/01/2010:23:35-06/09/2010:01:57	RELS	From-To: 06/02/2010-06/08/2010	BOND	7/2= 3.50 days
Arr/Chg:	5/1 MAKING A FALSE OR MISLEADING STATEMENT TO A PUBLIC SERVANT	Court/Case: CA/235B10D				
2010004108	BRATCHER, DONALD	Book-Release: 02/27/2010:21:40-06/09/2010:00:46	RELS	From-To: 06/01/2010-06/08/2010	SRVD	8/1= 8.00 days
Arr/Chg:	1/1 DRIVE W/LICENSE SUSP OR REVOKED-1	Court/Case: MM/C13589A				
2010004786	BRUMLEY, RUSSELL	Book-Release: 03/09/2010:00:35	OUT	From-To: 06/01/2010-06/13/2010	NSEN	13/1=13.00 days
Arr/Chg:	6/1 POSSESSION OF STOLEN PROPERTY 3	Court/Case: MM/C11107A				
2010008639	DAIGLE, NINA	Book-Release: 05/04/2010:13:06-06/03/2010:15:01	RELS	From-To: 06/01/2010-06/02/2010	COOP	2/2= 1.00 days
Arr/Chg:	2/1 THEFT - THIRD DEGREE	Court/Case: MM/C9399A				
2010012116	DELUZIO, CONNIE	Book-Release: 06/23/2010:10:14-06/28/2010:14:12	RELS	From-To: 06/23/2010-06/28/2010	BOND	6/3= 2.00 days
Arr/Chg:	3/1 DOMESTIC VIOLENCE - COURT ORDER VIOLATES ORDER	Court/Case: MM/C12470A				
2010012391	DELUZIO, CONNIE	Book-Release: 06/26/2010:17:00-06/28/2010:21:28	RELS	From-To: 06/26/2010-06/28/2010	RLSE	3/1= 3.00 days
Arr/Chg:	1/1 FAILURE TO TO OBEY OFFICER OR TO COMPLY WITH RCW 46.61.021[3]	Court/Case: MM/C13925A				
2010009893	DEVEAU, CODY	Book-Release: 05/20/2010:22:09	IN	From-To: 06/01/2010-06/05/2010	SRVD	5/1= 5.00 days
Arr/Chg:	1/1 DRIVING UNDER THE INFLUENCE	Court/Case: MM/C14144A				
Arr/Chg:	1/1 DRIVING UNDER THE INFLUENCE	Court/Case: MM/C14144A				
2010005793	DUPERRE, RICHIE	Book-Release: 03/23/2010:12:14-06/06/2010:00:43	RELS	From-To: 06/01/2010-06/13/2010	RLSE	13/1=13.00 days
Arr/Chg:	1/1 ASSAULT 4 - DV; DOMESTIC VIOLENCE	Court/Case: UNK/C12069A				
2010010251	FEATHERSTON, THERESA	Book-Release: 05/26/2010:01:13-06/02/2010:18:13	RELS	From-To: 06/01/2010-06/02/2010	RLSE	2/1= 2.00 days
Arr/Chg:	1/1 DRIVE W/LICENSE SUSP OR REVOKED-1	Court/Case: MM/C8101A				
2010012381	FERREE, MATHEW	Book-Release: 06/26/2010:13:01	IN	From-To: 06/26/2010-06/30/2010		5/3= 1.67 days
Arr/Chg:	3/1 THEFT - THIRD DEGREE	Court/Case: MM/C13432A				
2010012386	HEFNER, WILLIAM	Book-Release: 06/26/2010:15:02-06/30/2010:12:41	RELS	From-To: 06/26/2010-06/30/2010	RLSE	5/1= 5.00 days
Arr/Chg:	1/1 URINATING IN PUBLIC	Court/Case: MM/C13924A				
2010011875	HINE, TED	Book-Release: 06/18/2010:23:23-06/23/2010:10:09	RELS	From-To: 06/19/2010-06/23/2010	BOND	5/1= 5.00 days
Arr/Chg:	1/1 DOMESTIC VIOLENCE - COURT ORDER VIOLATES ORDER	Court/Case: MM/C14353A				
2010012316	KADE, TORY	Book-Release: 06/25/2010:15:25	IN	From-To: 06/25/2010-06/30/2010	COOP	6/2= 3.00 days
Arr/Chg:	2/1 RESISTING ARREST	Court/Case: MM/C13451A				
2010012278	KNOLL, GLENN	Book-Release: 06/25/2010:02:30-06/30/2010:12:41	RELS	From-To: 06/25/2010-06/30/2010	RLSE	6/1= 6.00 days
Arr/Chg:	1/1 THEFT - THIRD DEGREE	Court/Case: MM/XY0242988				
2010010959	MANSFIELD, TERRY	Book-Release: 06/05/2010:12:36-06/09/2010:19:27	RELS	From-To: 06/05/2010-06/09/2010	RLSE	5/1= 5.00 days
Arr/Chg:	1/1 DRIVING UNDER THE INFLUENCE	Court/Case: UNK/C11964A				
2010009415	MCCRACKEN, DOUGLAS	Book-Release: 05/14/2010:13:54-06/19/2010:06:56	RELS	From-To: 06/01/2010-06/18/2010	SRVD	18/1=18.00 days
Arr/Chg:	4/1 DRIVING UNDER THE INFLUENCE	Court/Case: MM/C7209A				

Appendix A - ODS Style Attributes

Attribute	Description	HTML	RTF	PDF
ABSTRACT=	Specify whether or not graph styles are used in CSS or LaTeX style files.	x		
ACTIVELINKCOLOR=	Specify the color for links that are active.	x	x	
ASIS=	Specify how to handle leading spaces and line breaks.	x	x	x
BACKGROUND=	Specify the color of the background of the table or graph	x	x	x
BACKGROUNDIMAGE=	Specify an image to use as the background.	x		
BODYSROLLBAR=	Specify whether or not to put a scrollbar in the frame that references the body file.	x		
BODYSIZE=	Specify the width of the frame that displays the body file in the HTML frame file.	x		
BORDERCOLOR	Specify the color of the border if the border is just one color.	x		x
BORDERCOLORDARK	Specify the darker color to use in a border that uses two colors to create a three-dimensional effect.	x		x
BORDERCOLORLIGHT	Specify the lighter color to use in a border that uses two colors to create a three-dimensional effect.	x		x
BORDERWIDTH	Specify the width of the border of the table.	x		x
BOTTOMMARGIN=	Specify the bottom margin for the document.	x	x	x
BULLETS=	Specify the string to use for bullets in the contents file.	x		
CELLHEIGHT=	Specify the height of the cell.	x	x	x
CELLPADDING=	Specify the amount of white space on each of the four sides of the text in a cell.	x	x	x
CELLSPACING=	Specify the thickness of the spacing between cells.	x	x	x
CELLWIDTH=	Specify the width of the cell.	x	x	x
CONTENTPOSITION=	Specify the position of the frames in the frame file that displays the contents and the page files.	x		
CONTENTSCROLLBAR=	Specify whether or not to put a scrollbar in the frames in the frame file that displays the contents and the page files.	x		
CONTENTSIZ=	Specify the width of the frames in the frame file that display the contents and the page files.	x		
CONTRASTCOLOR=	Specify the alternate colors for maps. The alternate colors are applied to the blocks on region areas in block maps.	x	x	x
DROPSHADOW=	Specify whether to use a drop shadow effect for text in a graph.	x	x	x
ENDCOLOR=	Specify the end color for a gradient effect in a graph.	x	x	x
FILLRULEWIDTH=	Cause a rule of the specified width to be placed into the space around the text (or entire cell if there is no text) where white space would otherwise appear.			x
FLYOVER=	Specify the text to show in a tool tip for the cell.	x		x
FONT_FACE=	Specify the font to use.	x	x	x
FONT_SIZE=	Specify the size of the font to use.	x	x	x
FONT_STYLE=	Specify the style of the font.	x	x	x
FONT_WEIGHT=	Specify the font weight.	x	x	x
FONT_WIDTH=	Specify the font width compared to the width of the usual design.	x	x	x
FONT=	Specify a font definition.	x	x	x
FOREGROUND=	Specify the color of text or data items	x	x	x
FRAME=	Specify the type of frame to use on an HTML table.	x	x	x
FRAMEBORDER=	Specify whether or not to put a border around the HTML frame for an HTML file.	x		
FRAMEBORDERWIDTH=	Specify the width of the border around the HTML frames for an HTML file.	x		
FRAMESPACING=	Specify the width of the space between HTML frames for HTML files.	x		
GRADIENT_DIRECTION=	Specify the direction of the gradient effect in either the X or Y axis direction to influence the graph background, legend background, charts, walls, floors, etc.	x	x	x
HREFTARGET=	Specify the window or frame in which to open the target of the link.	x		
HTMLCLASS=	Specify the name of the stylesheet class to use for the table or cell.	x		
HTMLCONTENTTYPE=	Provide the value of the content type for pages that you send directly to a web server rather than to a file.	x		
HTMLDOCTYPE=	Specify the entire doctype declaration for the HTML document, including the opening "<!DOCTYPE" and the closing ">".	x		
HTMLID=	Specify an ID for the table or cell.	x		
HTMLSTYLE=	Specify individual attributes and values for the table or cell.	x		
IMAGE=	Specify the image to appear in the background. This image can be positioned or tiled.	x	x	x
INDENT=	Set a numeric value to use as the indentation depth.		x	x
JUST=	Specify justification.	x		x
JUST=	Specify the image's horizontal positioning.	x	x	x
LEFTMARGIN=	Specify the left margin for the document.	x		x

Appendix A - ODS Style Attributes

Attribute	Description	HTML	RTF	PDF
LINESTYLE=	Specify the line type to use in a graph. You can use SAS/GRAPH line types -46.	x	x	x
LINETHICKNESS=	Specify the thickness (width) of a line that is part of a graph.	x	x	x
LINKCOLOR=	Specify the color for links that have not yet been visited.	x	x	x
LISTENTRYANCHOR=	Specify whether or not to make this entry in the table of contents a link to the body file.	x		
LISTENTRYDBLSPACE=	Specify whether or not to double space between entries in the table of contents.	x		
MARKERSIZE=	Specify the size of the symbol used to represent data values.	x	x	x
MARKERSYMBOL=	Specify the symbol used to represent data values.	x	x	x
NOBREAKSPACE=	Specify how to handle space characters.	x		x
OUTPUTHEIGHT=	Specify the height for graphics in the document.	x	x	x
OUTPUTWIDTH=	Specify the width of the table or of the graph or line thickness.	x	x	x
OVERHANGFACTOR=	Specify an upper limit for extending the width of the column.	x	x	x
PAGEBREAKHTML=	Specify HTML to place at page breaks.	x		
POSTHTML=	Specify the HTML code to place after the HTML table or cell.	x		
POSTIMAGE=	Specify an image to place after the table or cell.	x		x
POSTTEXT=	Specify text to place after the cell or table.	x	x	x
PREHTML=	Specify the HTML code to place before the HTML table or cell.	x		
PREIMAGE=	Specify an image to place before the table or cell.	x		x
PRETEXT=	Specify text to place before the cell or table.	x		x
PROTECTSPECIALCHARACTERS=	Determine how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted.	x	x	x
RIGHTMARGIN=	Specify the right margin for the document.	x		
RULES=	Specify the types of rules to use in a table.	x	x	x
STARTCOLOR=	Specify the start color for a gradient effect in a graph.	x	x	x
TAGATTR=	Specify text to insert in the HTML	x		
TOPMARGIN=	Specify the top margin for the document.	x	x	x
TRANSPARENCY=	Specify the level of transparency for a graph.	x	x	x
URL=	Specify a URL to link to.	x	x	x
VISITEDLINKCOLOR=	Specify the color for links the visited links.	x	x	
VJUST=	Specify vertical justification.	x	x	x
WATERMARK=	Specify whether or not to make the image that is specified by BACKGROUNDIMAGE= into a "watermark." A watermark appears in a fixed position as the window is scrolled.	x		

Using Styles in...

PROC REPORT *STYLE=[style attributes]*
can be placed on PROC REPORT, *STYLE(HEADER)=[style attributes]*
DEFINE or COMPUTE statements *STYLE(COLUMN)=[style attributes]*

PROC TABULATE *STYLE=[style attributes]*
can be placed on the CLASS,
VAR and CLASSLEV statements
or as a BOX= option value

PROC TABULATE *[STYLE=[style attributes]]* note extra brackets
as part of a TABLE definition

PROC PRINT *STYLE=[style attributes]*
can be placed on the PROC PRINT *STYLE(HEADER)=[style attributes]*
or VAR statements *STYLE(COLUMN)=[style attributes]*

Escape sequences *^S=[style attributes]* assuming ^ as ODS ESCAPECHAR