

Creating RTF-Output Using ODS and PROC TEMPLATE

Marcel Hoevenaars and Machteld Baljet, Blue Gum Data Analysis, Blackheath, NSW

ABSTRACT

The Rich Text Format (RTF) destination of the new SAS® Output Delivery System makes it possible to produce output that can be incorporated (almost) seamlessly into word processors, such as Microsoft Word®.

ODS RTF is easy to start with. Without much effort you can produce Microsoft Word tables, that you could only dream of in SAS version 6.

For people wanting more control over fonts, colours, titles, etcetera, there is the new procedure TEMPLATE that can be used to define style definitions that change the appearance of all output generated by SAS. Also there is the STYLE-option that can be used within reporting procedures such as REPORT. If that is still not enough, then you can use RTF code to further format your reports.

This paper gives an overview of the possibilities of the ODS RTF destination, PROC TEMPLATE and PROC REPORT. We hope to provide you with the tools and knowledge you need to start producing RTF output, as well as to adapt the standard output to virtually any requirement that you can have.

INTRODUCTION

Up to version 6.12 of the SAS System, output from all non-graphical procedures, including reporting procedures such as PROC REPORT and TABULATE, was designed for a traditional line-printer. Output was limited to mono-spaced fonts, like Courier, and looked very boring and old-fashioned. Incorporation in other systems such as word processors was cumbersome and did not improve the looks.

A typical example of SAS version 6.12 output from PROC REPORT is:

```
-----+-----+-----+-----+-----+-----+
|                                     Heart rate |
|                                     Week         |
| Subject   1     2     3     4   Mean |
|-----+-----+-----+-----+-----+
| 001      | 67| 72| 70| 88| 74.3|
|-----+-----+-----+-----+-----+
| 002      |110| 98| 87|103| 99.5|
|-----+-----+-----+-----+-----+
| 003      | 87| 85| 89| 81| 85.5|
|=====+=====+=====+=====+=====+
|          |88.0|85.0|82.0|90.7| 86.4|
|-----+-----+-----+-----+-----+

```

Since SAS version 7 output can be written easier, better and more flexible to different destinations with the SAS Output Delivery System (ODS). Since the introduction of SAS version 8.1 we can write the output as Rich Text Format (RTF), which supports many fonts and styles and can be

directly incorporated into word processors such as Microsoft Word.

So let us create the same report as in the first example with ODS RTF. It is easy!

The following code:

```
ODS RTF FILE = "Example2.RTF";

... TITLE and FOOTNOTE statements...

PROC REPORT ..;
...statements ...
RUN;

ODS RTF CLOSE;
```

produces the following output:

Example 2 Producing RTF with defaults

	Heart rate				
	Week				
Subject	1	2	3	4	Mean
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
	88.0	85.0	82.0	90.7	86.4

File: Example2.RTF

That looks promising, doesn't it? At least it is much better than the report we produced in the first example.

But ...

- ... can we change all fonts from Times Roman to Arial?
- ... can we change the background colour of the table heading to white or another colour?
- ... can we change the style of cells depending on their value (traffic lighting)?
- ... can we change the style of the title and have different styles for different titles?
- ... can we put "Page m of n" in the page header or footer?
- ... etcetera ...

The answer is: "Yes, you can!"

In this paper we give an overview of the possibilities provided by the SAS system version 8.2 to create reports in the RTF format, drawn from different sources and our experience from reporting data from clinical trials.

We try to equip you with the tools and knowledge to do all things mentioned above and a few things more as well.

We limit ourselves to the REPORT procedure, but in general the same techniques apply for the TABULATE and PRINT procedure.

Most of the time we spend on techniques to enhance the ODS RTF output. At the end we address promoting uniformity of output within a company (or department, or project).

Although some of the features we describe have been available since version 8.1 of the SAS system, in general this is a version 8.2 paper and we guarantee proper functioning only in version 8.2 or later.

ENHANCING ODS RTF OUTPUT

Enhancing ODS RTF output is not too difficult if you know how it works and were to start.

In this chapter we have a look at several ways to modify the default ODS RTF output as shown in the introduction. We start with using some specific options of the ODS RTF statement. We then give an overview of how to create and modify styles with PROC TEMPLATE and how to use them in connection with PROC REPORT. Finally we explain how to modify the appearance of titles and footnotes.

SOME SPECIFIC OPTIONS FOR ODS RTF

You can use any general ODS option with ODS RTF. For instance you can use the SELECT or EXCLUDE option to select respectively exclude output objects.

In this section we have a look at a few more specific options.

By default, ODS RTF puts the titles generated with the TITLE-statement in the header and the footnotes generated with the FOOTNOTE-statement in the footer of the RTF-document. This is convenient if the table spreads over more than one page, but it is quite a nuisance, if you want to incorporate the table into a document that has its own headers and footers.

Currently there is no way to change this behaviour. There is an undocumented, experimental option BODYTITLE (we found it on a FAQ page on the website of SAS Institute in France) that should force ODS RTF to write the titles and footnotes to the body, but we did not get this option to work properly and we do not recommend the use of this option.

Also by default, ODS RTF starts a new page, i.e. writes a hard page break, at the beginning of each

procedure, and whenever a procedure requests a new page, e.g. for every new BY-group.

The option STARTPAGE= effects this behaviour.

With the option STARTPAGE= you have better control of when ODS RTF starts a new page.

The syntax is:

```
ODS RTF ... STARTPAGE = value ... ;
```

The STARTPAGE= option can be set to the following values:

YES	Inserts a new page at the start of each procedure and when a procedure requests a new page. STARTPAGE = YES is the default.
NO	No new pages are inserted at the start of each procedure or when it is requested by the procedures.
NOW	Works in conjunction with STARTPAGE = NO and inserts one page break immediately after the call

ON can be used as an alias for YES; OFF can be used as an alias for NO.

Specifying STARTPAGE = NO also suppresses writing the titles and footnotes to the RTF output.

This problem can be easily solved:

Before the first procedure call specify:

```
ODS RTF ...STARTPAGE = NO ...;
ODS RTF STARTPAGE = NOW;
```

This approach suppresses all hard page breaks, but puts titles and footnotes in place.

As with some other ODS destinations, you can use the STYLE= option to select a style definition.

The syntax is:

```
ODS RTF ... STYLE = style-definition ... ;
```

style-definition must be a one or more level name of a style definition that exists in a template store that has been defined in the ODS path.

The ODS path will be explained later.

By default ODS RTF uses the RTF style definition, i.e. is Styles.RTF that is in the template store Sashelp.Tmplmst. This template store contains 16 other style definitions shipped with SAS. You can find their names via the Results window and then View → Templates.

The following code:

```
ODS RTF ... STYLE = Minimal ... ;
```

(Note that if the style definition you want to use is in the directory Styles in the template store, than you may omit the directory name Styles.)

Produces this output:

Producing RTF with STYLE = Minimal

Heart rate					
Week					
Subject	1	2	3	4	Mean
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
	88.0	85.0	82.0	90.7	86.4

File: Example3.RTF

If you are completely happy with one or more of these styles, you do not need to read any further.

STYLE ATTRIBUTES, STYLE ELEMENTS AND STYLE DEFINITIONS

To avoid confusion later on let us first define some terms.

STYLE ATTRIBUTE

A *style attribute* is a presentation feature. Style attributes determine the size, face, and weight of the type that is used, the colour of the foreground and background, and other such features.

An example of a style attribute with its value is `Font_face = "Arial, Verdana"`

In the next table we give a sample of style attributes that are relevant for RTF. Most of them will be covered in this paper.

Attribute	Description
Background	colour of the background
Font	font definition, consist of a combination of font face, font size, font style and font weight
Font_Face	font face
Font_Size	font size
Font_Style	font style (e.g. italic)
Font_Weight	font weight (e.g. bold)
Foreground	colour of the foreground
Just	(horizontal) justification
OutputWidth	width of the table
CellHeight	height of the cell. A row will have the height of the highest cell.
CellPadding	amount of white space on all four sides of text in a cell

Attribute	Description
PrelImage	Specify an image to be place before the table or cell.
PostImage	Specify an image to be place after the table or cell
PreText	Specify text to place before the table or cell
PostText	Specify text to place after the table or cell
LeftMargin	width of left margin
RightMargin	width of right margin
BottomMargin	width of bottom margin
TopMargin	width of top margin

A list of all predefined style attributes provided by SAS together with their possible values can be found in "the Complete Guide to the SAS Output Delivery System, PROC TEMPLATE, Syntax, Define Style".

A set of attributes with there respective values makes up a *style element*.

STYLE ELEMENT

A *style element* is a collection of style attributes and their values that can be applied to a particular part of the output.

Procedures may use different style elements in different parts of their output. For example, a procedure can use one style element for column headers and another for data.

An example of a style element is "Header". This style element effects the way column headers are presented. The style element "Header" may consist of the following style attributes:

`Font_face = "Arial, Verdana"`
`Font_weight = bold`
`Font_size = 12 Pt`
`Background = white`

A set of style elements makes up a *style definition*.

STYLE DEFINITION

A *style definition* is composed of style elements. It describes how to render the presentation aspects (colour, font face, font size, and so forth) of all output produced on an ODS destination. A style definition determines the overall look of the documents that use it. A style definition can be used for any ODS destination that supports the `STYLE=` option, this includes the ODS RTF destination.

An example of a style definition is the style definition RTF, the default style definition used by

ODS RTF. This style definition is a collection of over 100 style elements.

Only one style definition can be active per ODS destination at the time.

Creating and adapting style definitions is one of the things that can be done with PROC TEMPLATE. Explaining this functionality is part of this paper.

Other functionality of PROC TEMPLATE is defining table templates to render a SAS-dataset or an output object of a SAS procedure. This functionality lies outside the scope of this paper.

CREATING AND ADAPTING STYLE DEFINITIONS, STYLE ELEMENTS AND STYLE ATTRIBUTES

Style definitions, style elements and style attributes can be created or adapted using PROC TEMPLATE.

In the next part we spend some time on the ins and outs of PROC TEMPLATE, addressing the syntax and the concept of inheritance.

Before we start, let us first have a look at where output of PROC TEMPLATE is stored.

Style definitions and other things produced by PROC TEMPLATE are stored in so called template stores (files that are template stores have an extension ".sas7bitm"). Using the ODS PATH statement you can tell SAS which locations to use for reading from and writing to template stores.

For example:

```
LIBNAME mystyles "c:\MyStyles";  
ODS PATH mystyles.myRTFstyles(WRITE)  
sashelp.tmplmst(READ);
```

takes care that the style definitions are created in template store mystyles.myRTFstyles on directory c:\Mystyles and that styles used by ODS and PROC TEMPLATE are first looked up in that same template store and, if not found, in the template store sashelp.tmplmst, that has been shipped with SAS.

SYNTAX OF PROC TEMPLATE

Creating or adapting a style definition is done with the DEFINE STYLE statement in PROC TEMPLATE. In summary the syntax is:

```
PROC TEMPLATE;  
  DEFINE STYLE MyNewStyleDefintion;  
    <PARENT = ExistingStyleDefinition;>  
    <statements>  
  END;  
RUN;
```

Note that PROC TEMPLATE does not have any options and that an END statement must be the last statement in the definition.

The PARENT = statement is used to nominate an existing style definition, that you want your new style definition to inherit from. The concept of inheritance will be explained in more detail shortly.

A style definition consists of one or more style elements. A style element is created within a style definition block with a STYLE statement or a REPLACE statement:

```
STYLE MyStyleElement  
<FROM ExistingStyleElement>  
<"Comment"> /  
<Attribute1 = value1 ...>;  
REPLACE MyStyleElement  
<FROM ExistingStyleElement>  
<"Comment"> /  
<Attribute1 = value1 ...>;
```

If the attribute you want to define is not an attribute predefined in SAS, you have to put quotes around its name. For an overview of all predefined attributes see the "Complete Guide to the SAS Output Delivery System, PROC TEMPLATE".

The difference between STYLE and REPLACE all has to do with inheritance, so let us have a closer look at inheritance first.

INHERITANCE

When creating your own style definition, you do not have to start from scratch. You can take an existing style definition, e.g. the RTF style definition supplied by SAS and adapt it. That is what inheritance is all about.

PROC TEMPLATE supports two types of inheritance: style definition inheritance and style element inheritance.

Style definition inheritance is accomplished by using the PARENT= statement; style element inheritance by using the FROM keyword in the STYLE or REPLACE statement.

When you specify a parent for a style definition, all the style elements and attributes that are specified in the parent's definition are used in the new definition unless the new definition overrides them.

If you specify a style element after the FROM keyword in the STYLE or REPLACE statement, all attributes that are specified for the style element after the FROM keyword are used in the new style element unless you override them in the STYLE or REPLACE statement. The style element you use to inherit from must have been defined previously in the DEFINE STYLE block, or be part of the parent style definition.

The principle of inheritance is illustrated in the following example.

First we create a style definition My1stStyleDef from 'scratch". The style definition contains the style element Data:

```
PROC TEMPLATE;
  DEFINE STYLE styles.My1stStyleDef;
  STYLE Data /
    Font_face = Arial
    Font_size = 10 pt;
  END ;
  RUN ;
```

This results in the following style definition:

My1stStyleDef		
Style Element	Style Attribute	Value
Data	Font_face	Arial
	Font_size	10 pt

We then create another style definition My2ndStyleDef that has My1stStyleDef as a parent:

```
PROC TEMPLATE;
  DEFINE STYLE styles.My2ndStyleDef;
  PARENT = styles.My1stStyleDef;
```

This results in the following style definition:

My2ndStyleDef		
Style Element	Style Attribute	Value
Data	Font_face	Arial
	Font_size	10 pt

We now add a new style element DataEmphasis to My2ndStyleDef that inherits all style attributes from Data and add two style attributes:

```
STYLE DataEmphasis FROM Data/
  Foreground = White
  Background = GrayAA;
```

This results in the following style definition:

My2ndStyleDef		
Style Element	Style Attribute	Value
Data	Font_face	Arial
	Font_size	10 pt
DataEmphasis	Font_face	Arial
	Font_size	10 pt
	Foreground	White
	Background	GrayAA

Finally we extend My2ndStyleDef with style element Header that initially inherits all style attributes from the just created style element DataEmphasis and then overrides one of the style attributes:

```
STYLE header FROM DataEmphasis /
  Background = Black;
END;
RUN;
```

The final version of My2ndStyleDef is:

My2ndStyleDef		
Style Element	Style Attribute	Value
Data	Font_face	Arial
	Font_size	10 pt
DataEmphasis	Font_face	Arial
	Font_size	10 pt
	Foreground	White
	Background	GrayAA
Header	Font_face	Arial
	Font_size	10 pt
	Foreground	White
	Background	Black

Let us now create our report from previous example using our newly created style My2ndStyleDef:

```
ODS RTF FILE =... STYLE = My2ndStyleDef;

... TITLE and FOOTNOTE statements...

PROC REPORT ..;
  ...statements ...
RUN;

ODS RTF CLOSE;
```

This is what output looks like:

	Heart rate				
	Week				
	Subject	1	2	3	4
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
	88.0	85.0	82.0	90.7	86.4

The example above is actually a complicated way to getting there. It is only to illustrate how inheritance works.

The full power of inheritance is illustrated if we change the way My1stStyleDef is created. Instead of starting from scratch, we let the style definition inherit from the SAS supplied style RTF:

```
PROC TEMPLATE;
  DEFINE STYLE styles.My1sStyleDef;
  PARENT = styles.RTF;
  STYLE Data /
    Font_face = Arial
    Font_size = 10 pt;
  END ;
  RUN ;
```

If we create the style definition My2ndStyleDef as we did before and then create the report using this new version of the style definition we get the following output:

Heart rate					
Week					
Subject	1	2	3	4	Mean
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
	88.0	85.0	82.0	90.7	86.4

Now, all fonts in the table are Arial and the colour of the header and the summary line has been changed, but all other definitions from the default style definition, such as cell width, cell padding, have been inherited from the RTF style definition.

Inheritance is explained in much more detail in "the Complete Guide to the SAS Output Delivery System, PROC TEMPLATE, Concepts".

So now back to the difference between STYLE and REPLACE.

STYLE VERSUS REPLACE STATEMENT

You can use both the STYLE and the REPLACE statement to modify existing style elements. The STYLE statement makes the changes only to the style element that you specify, whereas the REPLACE statement makes the changes to the style element that you specify and to all the style elements that inherit from that element in the parent style definition.

In the following example the difference between STYLE and REPLACE is illustrated.

In this example we want to change all instances of the Times Roman font in the output into Arial. Instead of adapting all style elements, we want to change them all at once, in the style element Fonts that contains a list of all fonts referenced by all other style elements.

In the example we first modify the 'docfont' attribute in the style element Fonts by using the STYLE statement. We specify:

```
PROC TEMPLATE;
  DEFINE STYLE styles.MyStyleDef;
  PARENT = styles.rtf;
  STYLE fonts FROM fonts/
    'docFont' = ("Arial", 8 pt);
  END;
  RUN;
```

What happens here is that the attribute 'Docfont' of Fonts will now be Arial instead of Times Roman. That seems very nice, however it does

not change the actual output, because all style elements that inherit from the style Fonts and actual determine the output remain unchanged. What you need here is the REPLACE statement:

```
PROC TEMPLATE;
  DEFINE STYLE styles.MyStyleDef;
  PARENT = styles.rtf;
  REPLACE fonts /
    'docFont' = ("Arial", 8 pt);
  END;
  RUN;
```

This works better; some parts of the output are now in Arial, but other parts are still in Times Roman. Moreover, when we use the style definition we get a lot of warnings in the SASlog like:

```
WARNING: Could not locate style reference
'default.fonts("EmphasisFont")'.
```

These warnings appear because we replaced only the style attribute 'Docfont' of style element Fonts. All other style elements that were part of the parent definition Fonts are now undefined. Other style definitions that inherit them now cannot find them and that is what causes the warnings and Times Roman font still used in part of the output.

We need to replace all other attributes as well to get rid of the warnings and to get rid of Times Roman in the whole output:

```
PROC TEMPLATE;
  DEFINE STYLE styles.MyStyleDef;
  PARENT = styles.rtf;
  REPLACE fonts /
    'TitleFont2' = ("Arial", 10pt, Bold)
    'TitleFont' = ("Arial", 11pt, Bold)
    'StrongFont' = ("Arial", 8pt, Bold)
    'EmphasisFont' = ("Arial", 8pt, Italic)
    'FixedEmphasisFont' = ("Courier New,
      Courier", 7pt, Italic)
    'FixedStrongFont' = ("Courier New,
      Courier", 7pt, Bold)
    'FixedHeadingFont' = ("Courier New,
      Courier", 7pt, Bold)
    'BatchFixedFont' = ("SAS Monospace,
      Courier New, Courier", 6.7pt)
    'FixedFont' = ("Courier New,
      Courier", 7pt)
    'headingEmphasisFont' =
      ("Arial", 9pt, Bold)
    'headingFont' = ("Arial", 8pt, Bold)
    'docFont' = ("Arial", 8pt);
  END;
  RUN;
```

You can use the FROM keyword with the REPLACE statement, but this would not help you in getting rid of the warnings and wrong fonts in previous example. If you use the same style

element after the FROM keyword as the style element that you want to REPLACE, then the REPLACE statement replaces also the FROM style element in its entirety, so that you still have to define all style attributes. Rather confusing, isn't it?

You can use the REPLACE statement only in a style definition that has a parent.

More explanation and examples of STYLE and REPLACE can be found in "the Complete Guide to the SAS Output Delivery System, PROC TEMPLATE, Concepts" or in Chris Olinger's paper "Twisty Little Passages, All Alike – ODS Templates Exposed".

SOME STYLE ELEMENTS THAT YOU MAY WANT TO ADAPT

In the last section we illustrated the difference between the REPLACE and STYLE statement, by adapting the style element Fonts.

In this section we mention a few style elements that are the base of a lot of other style elements and that you may want to adapt to change the looks of your output "at the top" and in the most consequent way.

STYLE ELEMENT FONTS

Almost all other style elements reference, directly or via inheritance, to this list of fonts.

For instance, the style element Container is the style element that is used as the base for all container oriented elements (i.e. almost everything produced by PROC REPORT). For the SAS supplied style definition RTF in the style element Container the font name is not 'hard-coded', instead a reference is made to the style element Fonts:

```
style Container
  "Controls all container oriented elements." /
  font = Fonts('DocFont')
  ... ;
```

In the previous section we illustrated the impact of replacing this style element.

STYLE ELEMENT COLOR_LIST

Similar to the style element Fonts is the style element Color_list, in which all colours used in the style definition are mentioned.

This is how the color_list is defined for the SAS supplied RTF style definition:

```
Replace Color_list
  "Colors used in the style definition" /
  'link' = blue
  'bgH' = grayBB
  'fg' = black
  'bg' = white ;
```

Note that the style attribute names are within quotes, because they are not predefined style attributes. You may add you own style attribute names to the color_list and use them later on in your template definition, if you want to use more colours and grays than provided for in the SAS supplied RTF style definition..

An easy way to change the background colour of the header in PROC REPORT output is replacing the Color_list and changing the style attribute 'bgH':

```
PROC TEMPLATE;
  DEFINE STYLE styles.MyRTFStyleDef;
  PARENT = styles.rtf;
  REPLACE Color_list /
  'link' = blue
  'bgH' = white
  'fg' = black
  'bg' = white ;
  END;
RUN;
```

STYLE ELEMENT COLORS

The style element Color_list contains all colours used in the style definition. In the style element Colors colours from the style element Color_list are associated with parts of the SAS output.

Part of the definition of style element Colors for the SAS supplied RTF style definition is:

```
Replace colors
  "Colors used in the style definition" /
  ...
  'headerbg' = color_list('bgH')
  ...
  'docfg' = color_list('fg')
  'docbg' = color_list('bg');
```

Almost all other style elements reference, directly or via inheritance, to this list of style element.

For the SAS supplied style definition RTF the style element Container also references to style element Colors:

```
style Container
  "Controls all container oriented elements." /
  font = Fonts('DocFont')
  foreground = colors('docfg')
  background = colors('docbg')
  ....;
```

You can also change the background colour of the header in PROC REPORT by replacing the style attribute 'headerbg' in the style element Colors.

BODY (MARGINS)

Finally, another style element that you may want to change sooner or later is the style element Body. This is the best place to change the margins of the output RTF document.

```
PROC TEMPLATE;
  DEFINE mystyle;
    PARENT = styles.rtf;
    STYLE body FROM body
      leftmargin = 3 cm
      rightmargin = 2 cm
      topmargin = 3 cm
      bottommargin = 2 cm;
  END;
RUN;
```

OTHER STYLE ELEMENTS

All style elements that effect the RTF destination are summarized in http://www.sas.com/rnd/base/topics/templateFAQ/Template_rtf.html.

USING STYLE DEFINITIONS, STYLE ELEMENTS AND STYLE ATTRIBUTES

We spent quite some time on explaining how to create style definitions, style elements and style attributes. Now we are going to have a more detailed look at how we can use them to enhance the output from PROC REPORT and the titles and footnotes.

ENHANCING OUTPUT FROM PROC REPORT

Earlier we saw, that by changing the style element Header in a style definition, we could change the appearance of the header created by PROC REPORT. Let us now have a closer look at how style elements are linked to PROC REPORT output.

Output of PROC REPORT can be divided in six types of items, we will call them locations:

REPORT	Underlying table of the report
HEADER	Column headers and spanning headers
COLUMN	Cells of a column
LINES	Parts created with LINE statements in COMPUTE blocks
SUMMARY	Summary lines produced by a BREAK or an RBREAK statement
CALLDEF	Cells that are identified by a CALL DEFINE statement

There are five style elements that are attached to the six locations. Thus, these five style elements globally determine the appearance of the PROC REPORT output.

The following table links the style elements to the location:

Location	Default style element
REPORT	Table
HEADER	Header
COLUMN	Data
LINES	TableNoteContentCell
SUMMARY	DataEmphasis
CALLDEF	Data

You only need to change these five style elements in your style definition to change the complete look of your reports.

You can also change the appearance of PROC REPORT output in more detail by using style elements and style attributes in PROC REPORT statements.

The general approach for this is to use the STYLE-option in PROC REPORT statements:

```
STYLE(location) = style-element|
  {style-attribute(s)}
```

where *location* is one of the six types of items we identified above.

You can use the STYLE(*location*) = option in the statement that calls PROC REPORT. In that case, it effects all items of that location. For instance, if you specify:

```
PROC REPORT .. STYLE(Header) =
  {Background = White} ...;
```

then the background colour of all column headers and spanning headers will have a white background.

You may also specify the STYLE(*location*)= option in the PROC REPORT statement that effects a location to set style attributes or elements for an

individual item. The link between location and PROC REPORT statements is as follows:

HEADER	DEFINE statement
COLUMN	DEFINE statement
LINES	LINE statement in COMPUTE block
SUMMARY	BREAK or RBREAK statement
CALLDEF	CALL DEFINE statement

For example, to instruct ODS to create a grayAA background for the column header of variable Subject and leave the other column and spanning headers unaffected, you specify:

```
DEFINE subject / GROUP STYLE(Header) =
  {Background = GrayAA} ;
```

Individual STYLE-specifications override the STYLE-specification done for that location in the statement that calls PROC REPORT, if any.

For example, if you specify a white background for column and spanning header in the statement that calls PROC REPORT and later on you specify a grayAA background for one specific column header in the DEFINE statement for that column, then only that specific column header will be grayAA and all other headers will be white.

As an example of the STYLE= option in PROC REPORT, we illustrate the use of the style attribute PRETEXT to get a label 'Mean' in the first column of the summary line:

```
....
COMPUTE subject;
  IF subject = " THEN CALL DEFINE('_C1_',
    'STYLE', 'STYLE = DataEmphasis
      {PRETEXT = "Mean"}');
ENDCOMP;
....
```

Produces the wanted label:

	Heart rate				
	Week				
Subject	1	2	3	4	Mean
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
Mean	88.0	85.0	82.0	90.7	86.4

You can also provide a SAS format as a value for a style attribute and so make the appearance of individual cells data-dependent (traffic lighting).

As an example, we want all heart rates below 70 and above 100 to be bold in our report. First we define the following format:

```
PROC FORMAT;
  VALUE hilo
    LOW - 69 = 'Bold'
    70 - 99 = ''
    100 - HIGH = 'Bold';
```

Then we assign the just created format to the style attribute font_style in the STYLE(Column) option of the appropriate DEFINE statement; in this case the DEFINE statement for the ACROSS-variable Week:

```
PROC REPORT ... ;
  COLUMN subject heartRate, week;
  DEFINE subject / GROUP ;
  DEFINE week / ACROSS
    STYLE(Column) = {Font_style = hilo.};
  DEFINE heartRate / MEAN;
  ... statements ...
RUN;
```

And there we are:

	Heart rate				
	Week				
Subject	1	2	3	4	Mean
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
	88.0	85.0	82.0	90.7	86.4

ENHANCING TITLES AND FOOTNOTES

In previous sections we have improved, or if you want adapted, the output produced by PROC REPORT in several ways, but we have not done much with the titles and footnotes yet.

There are four places where you can control the appearance of SAS system titles and footnotes. They are, in order of precedence:

1. The active style definition
2. Options specified in the Title/Footnote statement
3. Inline formatting with control codes that can be processed by ODS RTF
4. Using RTF codes

Controlling the appearance of titles and footnotes in the active style definition can be done by using PROC TEMPLATE for defining or modifying the style elements that effect the appearance of titles and footnotes. These are:

TitleAndNoteContainer	Renders titles, footnotes, page numbers, bylines etc.
TitlesAndFooter	Renders both titles and footnotes
SystemTitle	Renders titles
SystemFooter	Renders footnotes

For instance, if you want to centre your titles, then you change the style element SystemTitle. How to change style elements with PROC TEMPLATE was explained previously.

You cannot control the appearance of separate titles or footnotes via the style definition.

Using options in the Title/Footnote statement is the simplest way to change the appearance of individual titles and footnotes. The following options can be used:

BOLD	Bolds the text
ITALIC	Makes text italic
COLOR	Specifies the colour
FONT	Specifies the font
HEIGHT	Specifies the height in point size
JUSTIFY	justifies LEFT, RIGHT or CENTER
LINK	Specifies the hyperlink to use

For example, the following titles and footnote statements:

```
TITLE1 FONT="Arial" BOLD HEIGHT=12 pt
      "Example 7";
TITLE2 FONT="Arial" HEIGHT=10 pt
      "Adapting titles and footnotes";
FOOTNOTE JUSTIFY=RIGHT HEIGHT=7 pt
      "File: Example6.RTF";
```

together with the SAS supplied style definition RTF generate the following output when used with PROC REPORT:

Example 8

Adapting titles and footnotes

	Heart rate				
	Week				
Subject	1	2	3	4	Mean
001	67	72	70	88	74.3
002	110	98	87	103	99.5
003	87	85	89	81	85.5
	88.0	85.0	82.0	90.7	86.4

File: Example8.RTF

Once you switch on BOLD or ITALIC you cannot turn it off within the same statement.

You have still more control over your titles and footnotes by inserting formatting control codes in the your TITLE and FOOTNOTE statements.

With this functionality you can put formatting control codes within the text of data, labels, titles and footnote to tell ODS RTF to use style attributes.

A sample of the formatting control codes that can be used is:

```
S={style-element|style-attribute(s)}
  gives you full control over the appearance of
  every title or footnote
S={ }
  resets previous S=
{super text}
  puts specified text in superscript
{sub text}
  puts specified text in subscript
```

Watch out!

Would you believe it? These codes are case-sensitive! Use case as given in table above!

These formatting control codes must be preceded by an escape character ('03'x), or a character that has been nominated as an alias for an escape character.

To nominate an alias to use as an escape character you can use the ODS ESCAPECHAR = statement. For example, if you specify:

```
ODS ESCAPECHAR = '\';
```

then '\ will be interpreted as an escape character. Note that you specify ODS ESCAPECHAR= in a separate ODS statement (not with ODS RTF). The ODS ESCAPECHAR effects all active ODS destinations, except LISTING and OUTPUT.

You can nominate every character as an escape character. However, the character will be used throughout the whole output, i.e. not only within the titles and footnotes sections, as an escape character, so be careful with your choice. It is time for an example.

The formatting control code "S" is very versatile in titles and footnotes, because you can specify every style attribute that effect cells. We illustrate the use of this formatting control by generating a logo in the header via a TITLE statement and the style attribute PrelImage:

```
ODS ESCAPECHAR = '~';
TITLE1 JUSTIFY = RIGHT "~S={PrelImage =
  'logo.jpg'}";
```

produces the following title section in output:

Example 9

Adapting titles and footnotes



TitleAndNoteContainer	To change the appearance of titles and footnotes
TitlesAndFooters	
SystemTitle	
SystemFooters	
Table	To change the style elements that are used by default by proc report
Header	
Data	
TableNoteContentCell	
DataEmphasis	
Body	To set the margins within the document
Fonts	To change the fonts and colours used throughout the style definition "once and for all"
Color_list	
Colors	

You can also add your own style elements to the style definition, e.g. a style element FirstColumn that renders data in bold. This element is not automatically used by PROC REPORT, but should be assigned to STYLE(Column)= in the DEFINE statement for the first column. Or define a style for all titles except the first, or the last footnote, as a source note.

Tip: If you want different margins for portrait pages than for landscape pages you need two style definitions. Create you portrait style definition first and use this as a parent for the landscape style definition. You only need to override the margin style attributes.

Note, that you have to specify OPTIONS PORTRAIT|LANDSCAPE to actually change the orientation in the RTF document.

You may also want to make different style definitions for different occasions or projects.

AVOID HARD CODING OF FONT NAMES, COLOURS ETC.

Use style elements or style attribute references instead.

Although it is tempting to use the much easier

```
TITLE FONT = Courier "Fixed font title";
```

It is better practice to specify:

```
ODS ESCAPECHAR = \';
TITLE "\S = {font = Fonts('FixedFont')} Fixed font title";
```

If the company decides to use another fixed font, with the first method you have to change the company style definition as well as your SAS program, with the second method, changing the company style definition is enough.

MAKE THE COMPANY STYLE DEFINITIONS ACCESSIBLE

Make sure that the SAS users within your company or department can access the style definitions.

The best way to accomplish this is by assigning a libref to the directory that contains the style definition and putting an appropriate ODS PATH statement in the AUTOEXEC.SAS file of the users.

Provide some central macros. For example create a macro that calls ODS RTF with the correct style definition.

EDUCATE THE USERS

Make sure that the users are aware of the company style definitions, that they know how to access them and know how to use them.

CONCLUSION

We did not have the time to ponder on the drawbacks of ODS RTF. There are of course a few. For instance, there is no possibility to merge empty cells in headers and columns that represent an ORDER or GROUP variable. Also there are limitations that are inherent to the RTF table functionality, e.g. only the table header is repeated when a page break occurs and not the values of ORDER and GROUP variables. And we saw already the inflexibility of where titles and footnotes are written.

However, we feel that the output quality, ease and flexibility of the new ODS RTF feature largely outweigh these drawbacks.

We hope that this paper has given you the knowledge and inspiration to start using this feature or, if you already did, to get even more out of it.

REFERENCES

Olinger, Chris, "Twisty Little Passages, All Alike – ODS Templates Exposed", SUGI 24, SAS Institute Inc., Cary, NC

Hull, Bob, "Now There Is an Easy Way to Get to Word, Just Use PROC TEMPLATE, PROC REPORT, and ODS RTF", SUGI 25, Synteract, Inc, Encinitas, CA

McNeill, Sandy, "Changes & Enhancements for ODS by Example (through Version 8.2)", SUGI 25, SAS Institute Inc., Cary, NC

SAS Institute, Inc., "the Complete Guide to the SAS Output Delivery System", SAS Institute Inc., Cary, NC

<http://www.sas.com/rnd/base/early-access/odsdoc2/sashtml/tw5195/index.htm>

SAS Institute, Inc., "The TEMPLATE FAQ", SAS Institute Inc., Cary, NC

http://www.sas.com/rnd/base/topics/templateFAQ/Template_rtf.html

Sagnier, Yves," Rich Text Format (RTF) Version 1.5 Specification", CENA, France

<http://www.cena.dgac.fr/~sagnier/info/formats/rtf/rtfspe15.htm>

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:



Marcel Hoevenaars



Machteld Baljet

Blue Gum Data Analysis
15 Radiance Avenue
Blackheath, NSW, 2785

Phone: 02-47876206

E-mail: info@bluegumdata.com.au

Website: <http://www.bluegumdata.com.au>