Paper 3432-2019

The Incredible, Accessible Report: Creating PDF Reports that Meet Compliance Standards in SAS® 9.4M6

Greg Kraus, SAS Institute, Inc.

ABSTRACT

SAS® 9.4M6 enables SAS® programmers to create PDF reports that fully meet the Web Content Accessibility Guidelines 2.0 (WCAG 2.0) conformance requirements. These are the guidelines that government and industry use to determine whether electronically produced output is usable by people with disabilities. With the accessibility features in SAS 9.4M6, it is possible to create reports that require zero post-processing remediation work, thus saving you time and money. By using the PRINT, TABULATE, and REPORT procedures, and the ODS Report Writing Interface (RWI), SAS 9.4M6 can prompt you to address certain accessibility problems detected in your code, create tables of data that are structured to be fully accessible to users, and add alternative text for images inserted into your reports. This paper demonstrates how to use these reporting procedures to create accessible PDF reports.

INTRODUCTION

PDF files are designed primarily to define an exact visual layout for a document that is the same regardless of how the document is viewed—desktop computer, mobile device, or printed. These visual layouts do not convey the necessary information for all users to be able to fully understand the contents of the document. For example, screen reader users can understand neither an image or a graph presented in the document, nor can they understand the relationships between table cells and table headers based simply on how they are drawn in the document.

In order for PDF documents to be accessible to all users, all of the content in the PDF must be tagged to indicate what each item is and how it relates to other items. When you create PDF documents, this tagging process is almost always a secondary process that must occur after the visual presentation has been defined. However, SAS 9.4M6 now provides the ability to create PDF files with the tagging structure necessary for you to create highly customizable reports that are fully accessible to all users without any post processing.

In SAS 9.4M6, the mechanics of creating the tags within the document are automated for you and have been improved since the addition of the ACCESSIBLE option in the ODS PDF statement in SAS 9.4M5. Depending on the type of content in your document, there might be additional information that you need to provide to ensure that the tags are set correctly. For example, if you insert a graph, you must provide a description for the graph, but SAS generates the tagging structure for you within the PDF to convey that information to all users, such as screen reader users.

The rest of this paper details how to effectively use the PDF accessibility features in the Output Delivery System (ODS) to create reports that meet compliance guidelines such as WCAG 2.0 Level AA and PDF/UA. These guidelines are used as the basis for accessibility requirements around the world by government and industry.

This paper outlines how to use ODS reporting procedures to create accessible PDF documents using the following Base SAS® language features:

- PROC PRINT
- PROC TABULATE
- PROC REPORT

- RWI
- PROC ODSTEXT

PDF AND ODS LAYOUT

All of the following techniques work with ODS LAYOUT, both gridded and absolute. They also work even if no explicit ODS layout is defined.

DOCUMENT METADATA

When you create a PDF document, one piece of metadata you need to supply is the title of the document. The title is not necessarily the same as the file name, so you must explicitly specify it. You must supply this information, along with other helpful metadata, when you first open the ODS PDF destination:

```
ods pdf style=pearl file='report.pdf' accessible
   title="Cloud Rambler Shoe Sales"
   author="John Cloudwalker"
   subject="End of Year Shoes Sales Report";
```

SYSTEM OPTION: ACCESSIBLEPDF

In order to create a PDF with the necessary tagging information, you must indicate that the PDF should contain accessibility tags. There are two different ways to do this. In SAS 9.4M5 and SAS 9.4M6, you can add the ACCESSIBLE option in the ODS PDF statement:

```
ods pdf style=pearl FILE="report.pdf" accessible;
```

Starting in SAS 9.4M6, in addition to the previous technique, you can alternatively set a system option to make all PDF output contain accessibility tags:

```
options accessiblepdf;
```

Once this option is set, all subsequent new PDF files will contain accessibility tags, even if the ACCESSIBLE option is absent from the ODS PDF file= statement.

If you are using SAS® Enterprise Guide® or SAS® Studio, there are additional options for ensuring that all PDF documents contain accessibility tags.

- In SAS Enterprise Guide, select **Solutions > Accessories > Registry Editor**. In the Registry, select **SAS_REGISTRY > ODS > DESTINATIONS > PRINTER > PDF**. Double-click or press Enter on the Accessible key and change the value to On.
- In SAS Studio, select **Preferences > Results**, and then select **Enable accessible PDF option**

If you use ods pdf startpage=now, you still must specify the ACCESSIBLE option in the statement:

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

SYSTEM OPTION: ACCESSIBLECHECK

Starting in SAS 9.4M6, programmers have the ability to automatically check their code for accessibility problems in the generated output as they run their programs. Any accessibility problems that are detected are written to the log as WARNING messages. To initiate these checks, enable the ACCESSIBLECHECK system option:

```
options accessiblecheck;
```

This option does not guarantee that all accessibility problems will be detected. It simply checks for items that can be determined automatically without manual, human inspection. This option checks for the following two conditions:

- Images that do not have descriptions
- Tables from the PRINT, TABULATE, and REPORT procedures, and from RWI that are constructed in ways known to cause accessibility problems

The following code snippet results in the Warning message shown in Figure 1 being written to the SAS log:

```
ods escapechar="^";
options accessiblecheck;
ods text="^s={preimage='low.png'} Low Performing Store";
```

```
72         ods escapechar="^";
73         options accessiblecheck;
74         ods text="^S={preimage='low.png'} Low Performing Store";
WARNING: Accessibility issue: The image does not contain a description.  Specify a text description for the image.
```

Figure 1. Warning Message in SAS Log

SYSTEM OPTION: ACCESSIBLETABLE

Starting in SAS 9.4M6, the new ACCESSIBLETABLE system option provides two pieces of functionality to make tables more accessible: creating visible table captions and altering the output of some previously inaccessible tables to make them accessible.

CREATING VISIBLE TABLE CAPTIONS

With the ACCESSIBLETABLE system option enabled, if the CONTENTS argument is defined in PROC TABULATE or PROC REPORT, that content appears as a visible table caption for the table. If the CAPTION argument is also defined, the CAPTION argument takes precedence as the visual caption. The following code snippet produces the table shown in Figure 2:

```
options accessibletable;
proc tabulate data=sashelp.shoes;
    where Product="Boot";
    class Product Region;
    var Stores Sales;
    label Product="Product";
    keylabel SUM=" ";
    table Region="" All="Worldwide", Sales="Sales"*f=dollar12.0 /
    contents="Boot Sales By Region";
run;
```

Boot Sales By Region

	Sales
Africa	$119,835
Asia	$62,708
Canada	$385,613
Central America/Caribbean	$190,743
Eastern Europe	$306,785
Middle East	$171,282
Pacific	$123,575
South America	$245,675
United States	$448,296
Western Europe	$296,031
Worldwide	$2,350,543

Figure 2. Visible Table Caption

PROC PRINT does not currently support printing visible captions.

CHANGING THE APPEARANCE OF TABLES

The default layouts of some table output in SAS are known to cause accessibility problems. The ACCESSIBLETABLE system option changes the way the table is visually and semantically structured in order to make it accessible. For example, with PROC TABULATE, if you have concatenated row variables and you are displaying their labels, the output creates a condition with the row headers that causes problems for screen reader users:

```
proc tabulate data=sashelp.shoes;
    class Region Subsidiary;
    var Stores;
    label Stores="Stores";
    keylabel sum=" ";
    table Region*(Subsidiary All="Total") All="Total Stores", Stores;
run;
```

In the table that is produced, the Region and repeated Subsidiary labels, highlighted in the following image (Figure 3), cause problems for screen readers trying to navigate this table. The highlighted cells cause screen readers to misread the table.

		Stores
Region	Subsidiary	
Africa	Addis Ababa	65.00
	Algiers	101.00
	Cairo	88.00
	Johannesburg	51.00
	Khartoum	71.00
	Kinshasa	56.00
	Luanda	30.00
	Nairobi	70.00
	Total	532.00
Asia	Subsidiary	
	Bangkok	5.00
	Seoul	59.00
	Tokyo	1.00
	Total	65.00

Figure 3. PROC TABULATE with Concatenated Row Variable Labels

Using the system option ACCESSIBLETABLE alters the way the table is constructed:

```
options accessibletable;
proc tabulate data=sashelp.shoes;
    class Region Subsidiary;
    var Stores;
    label Stores="Stores";
    keylabel sum=" ";
    table Region*(Subsidiary All="Total") All="Total Stores", Stores;
run;
```

With ACCESSIBLETABLE enabled, the concatenated variable names are not displayed in the table (Figure 4).

		Stores
Africa	Addis Ababa	65.00
	Algiers	101.00
	Cairo	88.00
	Johannesburg	51.00
	Khartoum	71.00
	Kinshasa	56.00
	Luanda	30.00
	Nairobi	70.00
	Total	532.00
Asia	Bangkok	5.00
	Seoul	59.00
	Tokyo	1.00
	Total	65.00

Figure 4. PROC TABULATE with Concatenated Row Variables with No Labels

DEFINING HEADERS IN TABLES

When data tables have column and row headers, those headers must be defined in the tagging structure as headings. This enables screen reader users to easily understand the labels for each data cell as they navigate through the table. The different report writing techniques handle table header creation in different ways.

PROC PRINT

With PROC PRINT, column headers are automatically defined by variable labels. Row headers are defined by one of the following two methods.

- Include the OBS column in the displayed table
- Define header rows with the ID statement

The following code demonstrates how to define row headers using the ID statement:

```
proc print data=sashelp.shoes noobs contents="Boot Sales in Africa";
    where Region="Africa" and Product="Boot";
    id Subsidiary;
    var Sales;
run;
```

This code produces a table (Figure 5) with the variables defined as column headers and the subsidiaries defined as row headers.

Subsidiary	Sales
Addis Ababa	$29,761
Algiers	$21,297
Cairo	$4,846
Johannesburg	$8,365
Khartoum	$19,282
Kinshasa	$13,921
Luanda	$6,081
Nairobi	$16,282

Figure 5. PROC PRINT with Column and Row Headers Defined

PROC TABULATE

PROC TABULATE automatically creates the correct column and row headers for you based on the variables used in the row and column dimensions.

PROC REPORT

PROC REPORT automatically defines column headers correctly. If you want row headers to be defined for your table, you must use either the ORDER or GROUP option in the DEFINE statement. Additionally, you must use the SPANROWS argument in PROC REPORT so that row headers that span multiple rows are assigned correctly:

```
proc report data=sashelp.shoes spanrows contents="United States Sales
Summaries";
    where Region="United States";
    column Subsidiary Product Sales;
```

```
    define Subsidiary/group 'Subsidiary';
    define Product/group;
    define Sales/analysis sum 'Sales';
run;
```

This code produces the following table (Figure 6) with column and row headers defined correctly.

United States Sales Summaries

Subsidiary	Product	Sales
Chicago	Boot	$82,483
	Men's Casual	$408,978
	Men's Dress	$261,607
	Sandal	$601
	Slipper	$329,235
	Sport Shoe	$17,347
	Women's Casual	$172,021
	Women's Dress	$293,313
Los Angeles	Boot	$85,932
	Men's Casual	$177,010
	Men's Dress	$147,670
	Sandal	$737
	Slipper	$98,866
	Sport Shoe	$16,307
	Women's Casual	$62,661
	Women's Dress	$148,129

Figure 6. PROC REPORT with Column and Row Headers Defined

RWI

The Report Writing Interface (RWI) in the DATA step provides a tremendous amount of flexibility in formatting tables. Since data in these tables is not laid out according to a predefined template, you must ensure that you explicitly set the column and row headers.

Column headers can be defined in one of two ways:

- In the FORMAT_CELL method, include a TYPE argument of H:

```
obj.format_cell(text: 'Number of Stores', type: 'H');
```

- Wrap all of the rows containing column headers with the ROW_START and ROW_END methods:

```
obj.head_start();
   obj.row_start();
      obj.format_cell(data: 'Subsidiary');
      obj.format_cell(data: 'Product');
      obj.format_cell(data: 'Sales');
   obj.row_end();
obj.head_end();
```

Row headers are assigned using the same FORMAT_CELL and TYPE technique as column headers.

```
obj.format_cell(text: Region, type: 'H');
```

PROVIDING TABLE DESCRIPTIONS: CAPTIONS AND SUMMARIES

There are two ways to provide captions and summaries of tables in PDF documents. There are differences between these two techniques and what impact they have on the PDF document, but they are both used for the same purpose. In fact, you likely will need to provide both a caption and a summary for each of your tables.

Providing captions for tables is like giving the table a visual title or label. This helps users refer to this table from other parts of the document when it is referenced. However, the caption is not always communicated to screen reader users in a way that is useful. The table summary is also a title or label for the table, but it is not printed on the page and is only available to screen reader users. However, it clearly communicates that label to the screen reader user.

When screen reader users encounter a table with a summary, the summary is read. This summary should provide a unique identifier of what this table displays. Sometimes it is as simple as "Table 1", Table 2", and so on. However, more helpful summaries briefly describe the contents of the table (for example, "Boot Sales By Region").

In order to ensure that all users will get the intended information, and to keep the visual captions and non-visual summaries synchronized, the following is recommended for PROC TABULATE and PROC REPORT:

1. Enable the ACCESSIBLETABLE option.
2. Use the CONTENTS= argument.
3. Provide text that summarizes the contents of the table (for example, 'Sales for Q4').

This will use the value of CONTENTS as both the visible caption and the non-visual summary.

The following is recommended for RWI:

1. Use the CAPTION argument for the visible caption.
2. Use the same text from the CAPTION argument in the DESCRIPTION argument for the non-visible summary.
3. For the preceding arguments, provide text that summarizes the contents of the table (for example, 'Sales for Q4').

The following code demonstrates how to create table captions and summaries using the CONTENTS argument in PROC TABULATE and the DESCRIPTION and CAPTION arguments in RWI:

```
options accessibletable;

/* Providing a summary using CONTENTS */
proc tabulate data=sashelp.shoes;
    where Product="Boot";
    class Product Region;
    var Stores Sales;
    label Product="Product";
    keylabel sum=" ";
    table Region="" All="Worldwide", Sales="Sales"*f=dollar12.0 /
    contents="Boot Sales By Region";
run;

/* Providing a summary using DESCRIPTION in RWI*/
obj.table_start(description: "Sales for Q4", caption: "Sales for Q4");
```

PROC PRINT does not create visual table captions, so use the CONTENTS= option to provide a non-visual table summary.

If you do not specify the ACCESSIBLETABLE option, but do use the CONTENTS argument in PROC TABULATE OR PROC REPORT, the non-visual table summary still gets created.

CREATING NEW TABLES FROM OTHER PROCEDURES

Some procedures produce table output that is not accessible. However, the output of these procedures can be repackaged using accessible table construction techniques in PROC PRINT, PROC REPORT, and RWI. For example, PROC UNIVARIATE displays results in a multi-column format. While this layout is useful for displaying a lot of information in a condensed area, it makes it difficult for screen reader users to understand which data cells belong with which row headers. The following code produces the table shown in Figure 7:

```
proc univariate data=sashelp.class;
   var Height Weight;
run;
```

Moments			
N	19	Sum Weights	19
Mean	62.3368421	Sum Observations	1184.4
Std Deviation	5.12707525	Variance	26.2869006
Skewness	-0.2596696	Kurtosis	-0.1389692
Uncorrected SS	74304.92	Corrected SS	473.164211
Coeff Variation	8.22479143	Std Error Mean	1.17623173

Figure 7. PROC UNIVARIATE Table with Multiple Columns

Instead of displaying the results in the original format, you can suppress the visual output with NOPRINT, send the output to a new data set using OUTTABLE, and then use PROC PRINT to display the results accessibly.

```
proc univariate data=sashelp.class outtable=work.classStats noprint;
   var Height Weight;
run;

proc print data=work.classStats label noobs;
    id _VAR_;
run;
```

This code produces the following accessible table (Figure 8).

Variable Name	Number of Nonmissing Observations	Number of Missing Observations	Sum of the Weights	Sum	Mean	Standard Deviation	Va
Height	19	0	19	1184.4	62.337	5.1271	
Weight	19	0	19	1900.5	100.026	22.7739	5

Figure 8. PROC PRINT Version of PROC UNIVARIATE Results

You could further customize this table by using more powerful techniques in PROC REPORT and RWI.

GRAPHS

There are multiple aspects to making graphs accessible in PDF files. Making a graph accessible includes not relying on color alone to display information, and also creating text descriptions that describe the data being presented.

USING MORE THAN COLOR TO PRESENT GRAPHS

If a graph contains a grouped variable, it is essential that users can differentiate between the groups without solely relying on color differences within the graph. This is accomplished through one of two methods depending on the type of plot you are creating.

For graphs with plot lines or plot markers the ATTRPRIORITY option on the ODS GRAPHICS statement needs to be set to NONE. This setting causes line patterns and markers to cycle through different shapes in addition to colors.

```
ods graphics / attrpriority=none;

proc sgplot data=sashelp.shoes description="Sales to Inventory Ratio for
Products in the United States";
    where Region="United States";
    scatter x=Inventory y=Sales / group=Product;
run;
```

This code produces a graph (shown in Figure 9) in which the plot markers use unique shapes in addition to unique colors to identify the group.

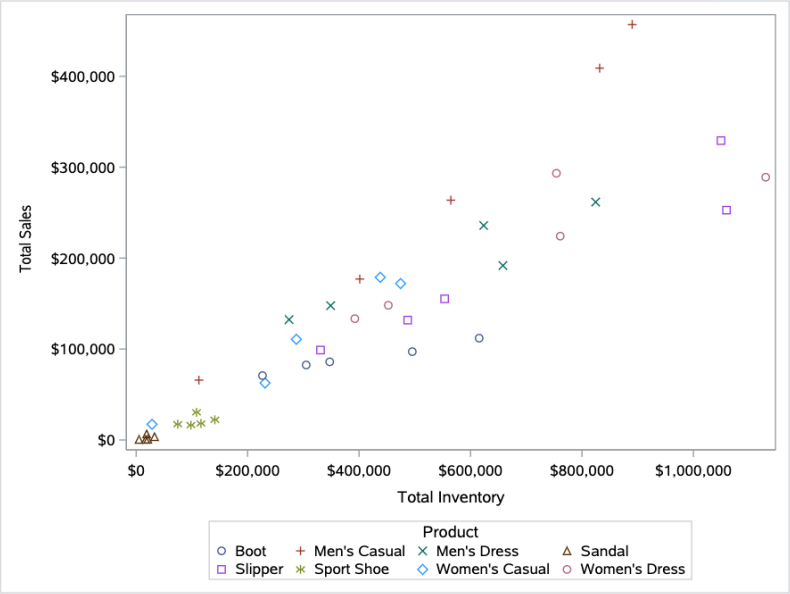

Figure 9. Grouped Scatter Plot with Different Marker Patterns

Additionally, for graphs with fill patterns, you also must specify the FILLPATTERN option in the graph type statement:

```
proc sgplot data=sashelp.shoes description="Product Sales By United States
Subsidiary";
  where region="United States";
  vbarparm category=Subsidiary response=Sales / group=Product FILLPATTERN;
run;
```

This code produces a graph (shown in Figure 10) in which the fill pattern varies along with the color for each group.

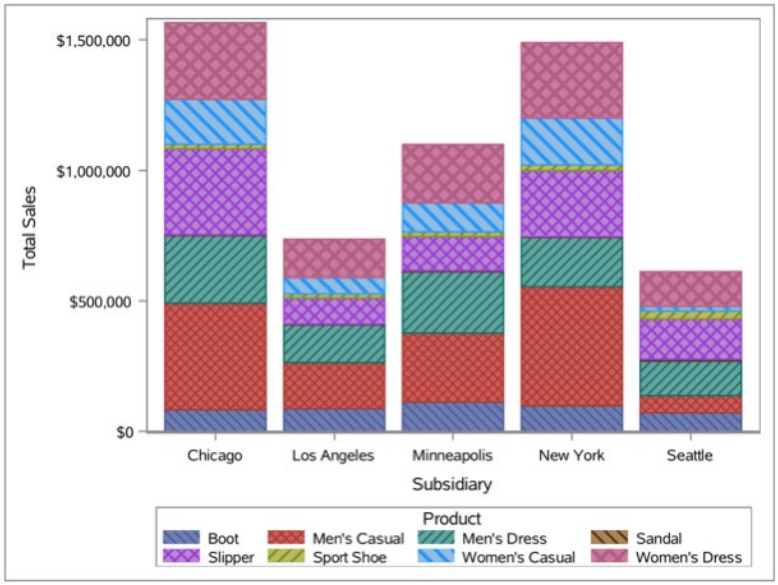

Figure 10. Grouped Bar Chart with Different Fill Patterns

CREATING DESCRIPTIONS OF GRAPHS

There are two aspects to creating descriptions of graphs:

- What makes a good description of a graph
- How to technically insert descriptions of graphs

Creating Good Descriptions of Graphs

The first aspect, creating good descriptions of graphs, is a complex subject. Graphs contain a lot of information. Describing all of that information adequately to people who cannot see the graph can be challenging. There are three strategies for dealing with this:

- Provide alternative text that briefly describes the data, including any analysis that would be apparent if you could see the graph (for example, "Average student weight increases linearly as student height increases").
- Provide alternative text that states what the graph is, but in the text within the page, ensure that a description of the data and analysis is present so that all users can read it (for example, "Graph showing miles per gallon vs. car weight"; in the page text, include descriptions of what the graph is showing).
- Provide alternative text that states what the graph is (as in the previous strategy), but in the text within the page, provide an accessible table containing the raw data used in the graph.

Adding Descriptions to Graphs

The second aspect, how to technically add descriptions to graphs in PDF documents, is very straightforward. The SGPLOT procedure enables you to add a DESCRIPTION argument. The text that you enter in the DESCRIPTION is read to the screen reader user.

```
proc sgplot data=sashelp.shoes description="Boot Sales by Region";
    where Product="Boot";
    vbar Region / response=Sales stat=Sum;
run;
```

HEADINGS, AND LISTS, AND TEXT, OH MY

A report seldom consists of just tables and graphs. Most of the time there is additional text in the report to show section headings and to provide a narrative to explain the data being presented. All of this information can be presented accessibly with PROC ODSTEXT.

HEADINGS

Whenever you have text that denotes the beginning of a section, you often use a larger and bolder font along with extra whitespace to help signify that text is labeling the content that follows it. None of this visual information is conveyed to screen reader users. To make sure all users can understand the content, mark it as an actual heading using PROC ODSTEXT and the HEADING statement (H for short). The following code demonstrates how to customize the way heading levels 1 and 2 appear in a document.

```
proc template;
  define style myPearl;
  parent=styles.pearl;

  /* H1 = Centered, larger, and bold */
  style Heading1 from Heading1 /
    fontsize=2.0em
    fontweight=bold
    textalign=center;

  /* H2 = Larger, bold, underlined, and whitespace on top and bottom */
  style Heading2 from Heading2 /
    fontsize=1.5em
    fontweight=bold
    textdecoration=underline
    margintop=0.5em
    marginbottom=0.5em;
  end;
run;

proc odstext contents="";
  h1 'My Report';
  h2 'Overview';
  p 'This report contains...';
  h2 'Summary';
  p 'In conclusion...';
run;
```

PLAIN TEXT

You can insert plain text by using PROC ODSTEXT and the P statement.

```
proc odstext;
  p 'Hello world';
run;
```

It can also be inserted using the shorter version.

```
ods text = "hello world";
```

LISTS

If you want to make a bulleted (unordered) list or a numbered (ordered) list, it is important not to just use space characters and an asterisk (*).

```
/* Do NOT do this - Creating a fake list using spaces and special
characters */
ods escapechar="^";
proc odstext;
  p '* Fruits';
  p '^_^_^_* Apple';
  p '^_^_^_* Banana';
  p '* Vegetables';
  p '^_^_^_* Carrot';
  p '^_^_^_* Turnip';
run;
```

You need to use either PROC ODSTEXT or PROC ODSLIST so that the items are structured as a proper list. In this way, screen reader users can understand the total number of items in a list and their relationship to each other.

```
/* Single Level Lists, using PROC ODSTEXT to make an ordered list */
proc odstext;
  list / start=1 style={liststyletype="decimal"};
    item 'Open SAS Studio';
    item 'Write your SAS program';
    item 'Run your SAS program';
  end;
run;

/* Nested Lists, using PROC ODSLIST to make an unordered list */
proc odslist;
  item;
    p 'Africa';
    list;
      item 'Addis Ababa';
      item 'Algiers';
    end;
  end;
  item;
    p 'Asia';
    list;
      item 'Bangkok';
      item 'Seoul';
    end;
  end;
run;
```

IMAGES OTHER THAN GRAPHS

If you want to add an image to your PDF that is something other than a graph generated from SGPLOT, SAS has multiple ways to add a description to an image inserted with the PREIMAGE and POSTIMAGE attributes. You can even choose to tell screen readers to ignore

an image when reading the PDF. Note that setting an image to be ignored should be done only if the image is decorative or is duplicating information that appears elsewhere. An image should not be marked to be ignored simply to avoid needing to provide an alternative text description for it.

PREIMAGE AND POSTIMAGE

The following techniques work wherever inline styles can be defined.

```
ods escapechar="^";

/* text description as its own an argument */
ods text="^{style inline [preimage='high.png' description='High
performance']}";

/* text description as an inline argument */
ods text="^S={preimage='high.png?desc=High performance'}";

/* decorative image (note the space character) */
ods text="^{style inline [preimage='clouds.png' description=' ']}";

/* decorative image (no spaces after the "?="") */
ods text="^S={preimage='clouds.png?desc='}";
```

RWI DATA STEP

```
/* text description as argument in RWI */
data _null_;
    dcl odsout obj();
    obj.image( file: "high.png", description: "High performance");
run;

/* decorative image (empty description option) */
data _null_;
    dcl odsout obj();
    obj.image( file: "clouds.png", description: "");
run;
```

BACKGROUND IMAGE (WATERMARK)

Another technique for marking an image to be ignored is to treat it as a background image. This is a useful technique when a decorative watermark is included in the document.

```
/**** Background image for cover page ****/
/* define a style for a page with a background image */
proc template;
  define style watermarkPearl;
    parent=styles.pearl;
    class body from document /
          background=_undef_
          /* Note: the background image needs to be sized to take up the
          whole page */
          backgroundimage="cloud-rambler-background-image.png";
    end;
run;

%let black = cx000000;

/* write the first page with the watermark */
```

```
ods pdf style=watermarkPearl file="report.pdf" accessible;
ods layout absolute x=0in y=0in width=7.5in;
ods region x=0in y=4.5in height=4in width=7.5in;
proc odstext contents="";
    h1 "Cloud Rambler Shoes Sales Report" /style=[just=c foreground=&black.
    font_size=28pt font_weight=bold];
run;
ods layout end;

/* use the non-watermark template for the rest of the report */
ods pdf style=pearl startpage=now accessible;
/* Rest of your report goes here */
ods pdf close;
```

This code produces the following page (Figure 11) in the PDF.

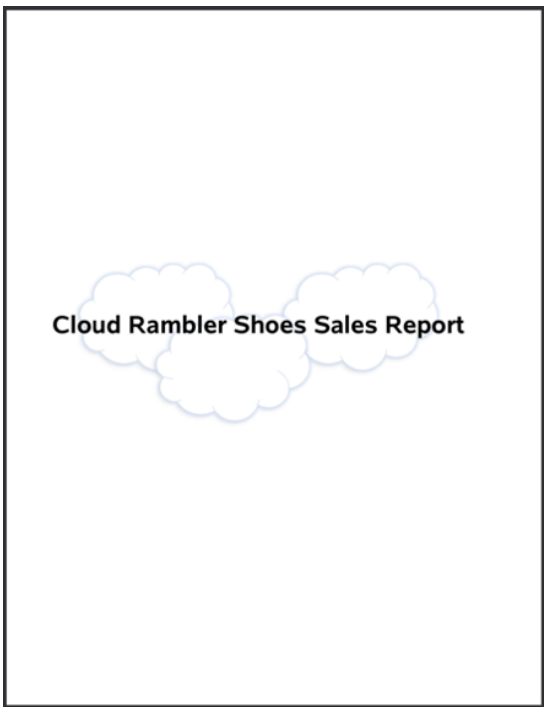

Figure 11. Watermark Image

ACROBAT ACCESSIBILITY CHECKER AND IMAGES IN TABLES

If you include images with alternative text descriptions in table cells, the Acrobat Accessibility Checker reports that those images fail the Nested Alternate Text test. The documentation for Adobe states that "Screen readers don't read the alternate text for nested elements. Therefore, don't apply alternate text to nested elements." **This is not correct and is a false positive from their checking tool**. This condition has been tested with JAWS 2018, JAWS 2019, NVDA 2018.4.1, and VoiceOver, and every screen reader reads the text alternative for the image that is nested in a table cell. If you want to disable this check in the Accessibility Checker, do the following:

1. Start the Full Check.
2. In the Accessibility Checker Options, go to the category **Alternate Text and Headings**.
3. Clear the **Alternate text that will never be read** check box.

ACCESSIBLE TRAFFICLIGHTING EXAMPLE

Let's take some of these principles and apply them to a real-world example—providing trafficlighting information in a table. When we trafficlight data in a table, we usually use background colors to indicate extra characteristics about the data, such as certain data being in a low or high range. The accessibility problem with this technique is if color is the only method by which we provide this information and if the user cannot see or discern different colors, then the user cannot receive this content.

One solution to this problem is to provide images that correspond to the trafficlighting levels in addition to background color changes. This solution enables people with color vision deficiencies to interpret the information. For example, if we have low, medium, and high price levels, we can use red, yellow, and green background colors, but we can also use images like the following:

- $
- $$
- $$$

Combined with the ability to add alternative text to these images, we can provide text descriptions for screen reader users so that they can also receive the information. In a case like this, the choice of alternative text is important. You want to make sure that you are describing the message the image is trying to convey, not a description of the image itself. In our example of low, medium, and high price levels, we would not want to make the descriptions of these images "one dollar sign", "two dollar signs", and "three dollar signs". We want to use descriptions like "low price", "medium price", and "high price", or whatever is appropriate to describe the trafficlighting levels.

Here is a code snippet showing the complete workflow for making accessible trafficlighting information from the sashelp.shoes data set to show which Subsidiaries had the highest and lowest sales per store for each product. This code uses PROC REPORT to create the final table and also produces a legend to explain what the trafficlighting icons mean:

```
/* Define the trafficlighting color codes */
%let BgGreen = cx80FF80;
%let BgYellow = cxFFEE80;
%let BgRed = cxFF8080;

/* Use PROC TABULATE just to calculate the totals for each product for each
region. Do not display the raw results */
ods select none;

proc tabulate data=sashelp.shoes out=work.regionProductSummaries;
  class Product Region;
  var Stores Sales Inventory Returns;
  table Region*Product All, Stores Sales Returns;
run;

/* restore output to the PDF destination */
ods select all;

/* calculate the sales per store for each product in each subsidiary */
data work.regionProductStats (keep=Region Product Stores_Sum Sales_Sum
Returns_Sum Ratio);
  set work.regionProductSummaries(where=(_TYPE_ NE "00"));
```

```
  length Ratio 8;
  format Sales_Sum dollar12.0 Returns_Sum dollar12.0 Ratio dollar12.0;
  Ratio = (Sales_Sum - Returns_Sum)/Stores_Sum;
run;

/* Sort the sales per store from lowest to highest */
proc sort data=work.regionProductStats out=work.regionProductStatsSorted;
  by ratio;
run;

/**** Trafficlighing levels for sales by region and product ****/
proc format;
  value salesLevel    low-10000 = &BgRed.
                      10000-20000 = &BgYellow.
                      20000-high = &BgGreen.;
  value salesImages   low-10000 = 'assets/low.png?desc=Low performing'
                      10000-20000 = 'assets/medium.png?desc=Medium
performing'
                      20000-high = 'assets/high.png?desc=High performing';
run;

proc report data=work.regionProductStatsSorted nowd
  contents="Product Sales Performance By Region" /* still keep the contents
so screen readers get this information */
  style(report)=[frame=void]
  style(column)=Data
  style(header)=Header
  spanrows;
  column Region Product Stores_Sum Sales_Sum Ratio;
  label Stores_Sum="Stores" Sales_Sum="Total Sales" Ratio="Sales Per
Store";
  define Region / group style(column)={width=2.2in};
  define Product / group style(column)={width=1.5in};
  define Stores_Sum / style(column)={width=0.75in};
  define Sales_Sum / style(column)={width=1in};
  define Ratio/ style(column)={width=1.5in backgroundcolor=salesLevel.
posttext="   " postimage=salesImages.};
run;

/* Display the legend for the table icons */
proc odstext contents="";
  h2 "Legend";
  p "^S={preimage='assets/low.png?desc=Low Performing'
backgroundcolor=&BgRed.}      Low performing subsidiary";
  p "^S={preimage='assets/medium.png?desc=Medium Performing'
backgroundcolor=&BgYellow.}   Medium performing subsidiary";
  p "^S={preimage='assets/high.png?desc=High Performing'
backgroundcolor=&BgGreen.} High performing subsidiary";
run;
```

PDF/UA COMPLIANCE—ACCESSIBILITY CONFORMANCE IDENTIFIER

There is a PDF accessibility standard called PDF/UA (Universal Access). One aspect of this standard is that the author can specify that a particular document has been checked for accessibility and has confirmed that it meets the PDF/UA requirements. You can set this identifier in your PDF document in the ODS PDF statement.

```
ODS PDF file="report.pdf" ACCESSIBLE ACCESSIBLE_IDENTIFIER;
```

This identifier does not add any extra accessibility support to your document. It simply enables the report author to confirm to the end user that the report meets the PDF/UA standard. This identifier should not be placed on a PDF file until you have completed the following steps:

1. Run the Accessibility Full Check within Adobe Acrobat.
2. Perform manual testing to ensure that the generated document is accessible.
3. Use a tool such as the PDF Accessibility Checker 3 to test against the testable PDF/UA criteria.

MANUALLY TESTING YOUR PDF FOR ACCESSIBILITY

SAS allows for a tremendous amount of flexibility in the type of output that you can generate. Because of this flexibility, you must always manually test the generated PDF for accessibility problems. These tests include the following:

- Using the accessibility testing tools in Adobe Acrobat. See the SAS Global Forum Paper on ODS PDF Accessibility in SAS® 9.4M5: Going Beyond the Basics to Create Advanced Accessible Reports, in the section "Checking For Accessibility—Knowing What You Need To Fix" (Kraus 2018).
- Using assistive technologies such as screen readers to ensure that all content can be read correctly.

SAMPLE PROGRAM

A SAS program and the resulting PDF demonstrating how to apply these principles in a complete report are available online at:

- SAS Program: http://support.sas.com/misc/accessibility/ods/shoes-report.sas
- Generated PDF: http://support.sas.com/misc/accessibility/ods/shoes-report.pdf

Image assets used in the code samples in this paper and in the refenced SAS Program can be found in the initial comments of the referenced SAS Program.

CONCLUSION

With the accessibility features in SAS 9.4M6, it is now possible to create highly customized and fully accessible reports in a PDF format, without having to perform any accessibility remediation. This author hopes that you are now hungry to upgrade to SAS 9.4M6 so that you can create your own incredible, accessible PDF reports.

REFERENCES

Adobe Systems Incorporated. *Accessibility Repair Workflow*. Accessed March 14, 2019. http://www.adobe.com/accessibility/products/acrobat/acrobat-pro-dc-pdf-accessibility-repair-workflow.html.

Adobe Systems Incorporated. 2012. "Adobe Acrobat XI Pro Accessibility Guide: Best Practices for PDF Accessibility." Available http://www.adobe.com/content/dam/acom/en/accessibility/products/acrobat/pdfs/acrobat-xi-pro-accessibility-best-practice-guide.pdf.

Adobe Systems Incorporated. *Using the Acrobat Pro DC Accessibility Checker*. Accessed March 14, 2019. http://www.adobe.com/accessibility/products/acrobat/using-acrobat-pro-accessibility-checker.html.

Adobe Systems Incorporated. *Training Resources*. Accessed March 14, 2019. http://www.adobe.com/accessibility/products/acrobat/training.html.

Kraus, Greg. 2018. "ODS PDF Accessibility in SAS 9.4M5: Going Beyond the Basics to Create Advanced Accessible Reports." *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. Available https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2124-2018.pdf.

PDF Association. The Matterhorn Protocol 1.02. Accessed March 14, 2019. https://www.pdfa.org/resource/the-matterhorn-protocol-1-02/.

PDF Association. "PDF/UA in a Nutshell." Accessed March 14, 2019. https://www.pdfa.org/wp-content/uploads/2013/08/PDFUA-in-a-Nutshell-PDFUA.pdf.

World Wide Web Consortium. "Web Content Accessibility Guidelines (WCAG) 2.0." Accessed March 14, 2019. https://www.w3.org/TR/WCAG20/.

RECOMMENDED READING

- Middleton, Woody. 2018. "PDF Accessibility: How SAS® 9.4M5 Enables Automatic Production of Accessible PDF Files" Proceedings of the SAS Global Forum 2018 Conference. Cary, NC: SAS Institute Inc. Available https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2129-2018.pdf/
- O'Connor, Daniel. 2017. "A Guru's Guide: Producing Section 508-Compliant Custom Reports with the Output Delivery System." Proceedings of the SAS Global Forum 2017 Conference. Cary, NC: SAS Institute Inc. Available https://support.sas.com/resources/papers/proceedings17/SAS0557-2017.pdf.
- SAS Institute Inc. 2019. Creating Accessible SAS® 9.4 Output Using ODS and ODS Graphics. Accessed March 14, 2019. Available https://go.documentation.sas.com/?docsetId=odsacoutput&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en.
- Walker, Glen. 2016. "A Guide to Section 508 Compliance Using SAS® 9.4 Output Delivery System (ODS)". Proceedings of the SAS Global Forum 2016 Conference. Cary, NC: SAS Institute Inc. Available https://support.sas.com/resources/papers/proceedings16/SAS5280-2016.pdf.
- Walker, Glen. 2017. "Tag, You're It! Creating Accessible (Tagged) PDF with SAS® 9.4 Output Delivery System." Proceedings of the SAS Global Forum 2017 Conference. Cary, NC: SAS Institute Inc. Available http://support.sas.com/resources/papers/proceedings17/SAS0483-2017.pdf.

ACKNOWLEDGMENTS

The author would like to thank the following SAS employees for carefully reviewing this paper:

- Bari Lawhorn

CONTACT INFORMATION

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

Greg Kraus
SAS Institute
100 SAS Campus Drive
Cary, NC 27513
+1 919-531-3153
Greg.Kraus@sas.com