

Staying Relevant in a Competitive World: Using the SAS® Output Delivery System to Enhance, Customize, and Render Reports

Chevell Parker, SAS Institute Inc.

ABSTRACT

Technology is always changing. To succeed in this ever-evolving landscape, organizations must embrace the change and look for ways to use it to their advantage. Even standard business tasks such as creating reports are affected by the rapid pace of technology. Reports are key to organizations and their customers. Therefore, it is imperative that organizations employ current technology to provide data in customized and meaningful reports across a variety of media. The SAS® Output Delivery System (ODS) gives you that edge by providing tools that enable you to package, present, and deliver report data in more meaningful ways, across the most popular desktop and mobile devices.

To begin, the paper illustrates how to modify styles in your reports using the ODS CSS style engine, which incorporates the use of cascading style sheets (CSS) and the ODS document object model (DOM). You also learn how you can use ODS to customize and generate reports in the body of email messages. Then the paper discusses methods for enhancing reports and rendering them in desktop and mobile browsers by using the HTML and HTML5 ODS destinations. To conclude, the paper demonstrates the use of selected SAS ODS destinations and features in practical, real-world applications.

INTRODUCTION

As times change, you can no longer depend on methods and techniques that were used to report and analyze data in the past. Today's customers expect more than ever before. To stay business relevant, especially in the area of business reports, you must use the latest reporting methods and techniques. If you are a SAS® software user, you are in a great position to meet those ever-changing customer needs.

Gone are the days when a single-listing report works as a medium for all of your customers. That would be like awakening after 20 years and looking for a phone booth or looking for your report on green-bar computer paper! Staying business relevant requires ways of delivering and reporting information that most benefit your customers.

This paper discusses the following methods for delivering and reporting your information in order to maintain business relevancy:

- customizing reports with cascading style sheets (CSSs)
- delivering formatted reports through email
- enhancing and rendering reports with HTML and HTML5
- using the new SAS® 9.4 features with the ODS Excel destination in practical applications

CUSTOMIZING REPORTS USING CASCADING STYLE SHEETS

Cascading style sheets (CSS) are an industry standard that are used to define the visual presentation for web pages. However, CSS is not just for web pages anymore! Now, you can use a CSS also to define the presentation of Adobe PDF files, Microsoft Excel worksheets, and RTF, and E-books. Creating styles with a CSS enables you to generate styles that can be applied globally in your reports, and this method is a worthy alternative to using the TEMPLATE procedure. The use of the CSSSTYLE= option and the ODS HTML destination's STYLESHEET= option should not be confused. The STYLESHEET= option, in combination with the URL= suboption in the HTML statement, simply passes the CSS style to the browser for rendering. On the other hand, when you use the CSSSTYLE= option, the CSS style is mapped to the PROC TEMPLATE elements and attributes.

Advantages of using a CSS for styles include the following:

- A CSS provides more dynamic styling capabilities than PROC TEMPLATE.
- CSS files offer greater portability because they are simply text files.
- Style properties that are used with ODS CSS styles are based on the W3C standard.
- Familiarity with CSS only requires a minimal learning curve.

The initial implementation of the CSS style engine in SAS® 9.2 was limited to using class selectors for targeting elements. This implementation simply converts the CSS class names to PROC TEMPLATE style elements and the CSS style properties to PROC TEMPLATE attributes. It also includes the ability to use the @Media rules, which enable you to set media types differently based on the output. Later, the CSS style engine was extended to include type or element selectors. In addition, the engine's capability was extended even further in SAS 9.4 to include several advanced methods of targeting elements (for example, using attribute and pseudo-class selectors), which are discussed later in this section. The complete feature set for the CSS style engine cannot be covered adequately in this paper. For a more detailed discussion of the features, see Smith (2013).

CASCADING STYLE SHEETS: THE BASICS

The anatomy of a CSS file includes one or more individual CSS rules that consist of selectors, properties, and values. The *selector* instructs the client about which elements should be matched when you generate styles. A *property* defines either how the style should look or its behavior, while a *value* is the argument that is passed.

The following example illustrates a basic rule that consists of a selector, two properties, and two values.

```
Selector {  
    property1: value1;  
    property2: value2;  
}
```

As shown above, the properties and values are enclosed in braces, with the property and its respective value separated by a colon.

THE ODS DOCUMENT OBJECT MODEL

It is easy to build the CSS that you need in order to add a style for HTML output because the HTML markup is visible. Before targeting various elements with selectors, you must first understand the ODS document object model (DOM), which is based on an HTML5 document tree and which enables you to target elements. The ODS DOM is similar to the HTML version of the document object model in some ways. However, the HTML DOM is an in-memory version of the HTML document. In SAS 9.4, you display the ODS DOM either by adding the DOM option in the specific ODS destination or by using the ODS TRACE DOM system option. Displaying the ODS DOM is necessary, initially, in order to view the elements that are available to be targeted by selectors. The process for displaying the DOM is explained in the next example. After you follow this process a few times, it becomes pretty familiar.

The following example code uses the DOM option in the ODS Excel destination (pre-production in SAS 9.4 TS1M1) to write the object model to the log. (**Note:** The ODS Excel destination is new in the first maintenance release of SAS 9.4 [TS1M1]. It is planned that the destination will be a production feature in SAS 9.4 TS1M3.)

```
ods excel file="c:\temp.xlsx" options(embedded_titles="yes"  
                                     sheet_name="Class_Selectors"  
                                     start_at="B2")  
                                     cssstyle="c:\class.css" dom;  
proc print data=sashelp.orsales(obs=10) noobs;  
    var Year Product_group quantity Profit;  
    format profit dollar.;  
    sum quantity profit;  
    title 'Profit Summary for Company ABC';  
run;  
ods excel close;
```

The information that is written to the log by this code is verbose, so only an abbreviated portion of the PRINT procedure output is shown below:

```

<!DOCTYPE html>
<html>
  <body class="body">
    <section id="idx" class="oo" proc="print">
      <table class="systitleandfootercontainer">
        <tr>
          <td class="systemtitle">Profit Summary for Company
            ABC</td>
        </tr>
      </table>
      <table class="table">
        <thead>
          <tr>
            <th class="header" type="char" index="1"
              name="year" >Year</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td class="data" type="num" index="1" name="year" >1999
            </td>
          </tr>
        </tbody>
      </table>
    </section>
  </body>
</html>

```

CASCADING STYLE SHEET SELECTORS

Pattern-matching rules are called *selectors* in CSS. Selectors determine which CSS rules apply to elements, making them one of the most important aspects of CSS. Selectors can range from very simple (for example, class and element selectors) to very advanced (for example, the pseudo-class and attribute selectors that are discussed later in this paper). Advanced selectors offer more flexibility, and with them, you can match almost every element or parts of the output.

The next sections discuss both basic and advanced selectors, and the sections use examples to illustrate how to apply these selectors to Microsoft Excel worksheets from within SAS.

Basic Selectors

This section discusses four types of basic selectors: class, element (type), ID, and universal. These selectors are the ones used most often in modifying the style of your output.

CSS Class Selectors

The *class selector* selects elements that are based on a specific class attribute. This type of selector is always preceded by a period. For example, the following code, which is from the ODS DOM, uses four class selectors (**.BODY**, **.SYSTEMTITLE**, **.HEADER**, and **.DATA**) to modify Excel output:

```
.body {
  background:white;
  font-family:arial;
}
.systemtitle {
  font-weight:bold;
  font-size:12pt;
  font-family:arial;
}
.header {
  background-color:teal;
  color:beige;
  font-family:arial;
  border:1px solid black;
}
.data {
  border:1px solid black;
  font-family:arial;
  background-color:beige;
}
```

Year	Product_Group	Quantity	Profit
1999	A-Team, Kids	288	\$4,980
1999	Bathing Suits, Kids	98	\$1,480
1999	Eclipse, Kid's Clothes	588	\$9,349
1999	Eclipse, Kid's Shoes	334	\$7,137
1999	Lucky Guy, Kids	303	\$7,163
1999	N.D. Gear, Kids	755	\$19,153
1999	Olssons, Kids	209	\$1,975
1999	Orion Kid's Clothes	14	\$289
1999	Osprey, Kids	454	\$7,335
1999	Tracker Kid's Clothes	1243	\$21,848
		4284	\$80,709

Output 1. Class Selectors Applied to a Microsoft Excel Worksheet

This example illustrates how you can target various elements using class selectors. The class selectors equate to (match) style element names in SAS.

In This Example:

- The **.BODY** class modifies the style of the worksheet body.
- The **.SYSTEMTITLE** class modifies the style of the worksheet title.
- The **.HEADER** and **.DATA** classes modify, respectively, the table headers and the table definition cells.

Adding class selectors to a file and referencing that file in the CSSSTYLE= option generates the output that is shown above in Output 1.

CSS Element (Type) Selectors

An *element (or type) selector* matches every element that is referred to in the ODS DOM with the corresponding element type name. The next example uses element (type) selectors to match the various parts of output that were modified with class selectors in the previous code example.

```
body {background-color:beige}

td.systemtitle {
  font-weight:bold;
  font-size:12pt;
  border-bottom:none;
}
```

(code continued)

```

th {
  background-color:brown;
  color:beige;
  font-weight:bold
  font-family:arial;
  border-bottom:1px solid
  black;
}
td {
  border-bottom:1px solid black;
  font-family:arial;
  font-weight:medium;
  font-size:9pt;
}

```

Year	Product_Group	Quantity	Profit
1999	A-Team, Kids	286	\$4,980
1999	Bathing Suits, Kids	98	\$1,480
1999	Eclipse, Kid's Clothes	588	\$9,349
1999	Eclipse, Kid's Shoes	334	\$7,137
1999	Lucky Guy, Kids	303	\$7,163
1999	N.D. Gear, Kids	755	\$19,153
1999	Olssons, Kids	209	\$1,975
1999	Orion Kid's Clothes	14	\$289
1999	Osprey, Kids	454	\$7,335
1999	Tracker Kid's Clothes	1243	\$21,848
		4284	\$80,709

Output 2. Applying Type Selectors in Microsoft Excel Output

In This Example:

- The **Body** element selector modifies the background color of the worksheet.
- Both the worksheet title and the data values use the **TD** element selector. Therefore, the selector for the title also includes the class name **.systemtitle** to further identify that the title should be targeted. Because of the way a CSS works, the styles that are listed in the **.systemtitle** class override any styles that appear later in the CSS.
- The **TD** selector modifies the data values.
- The **TH** element selector modifies the table headers.

CSS ID Selectors

An *ID selector* matches elements that have specific ID attribute values. An ID selector begins with a number sign (#), and the default ID value is **IDX**. Because the ID attribute must have a unique value, an ID selector can never match more than one element in the document. For example, if you have the default ID selector (**#IDX**) for one element, you must use a different ID selector (for example, **#IDX1**) for another element.

The following example demonstrates how you can use an ID selector to apply a style that modifies the font size of cells only in the first table in a worksheet:

```

<section id="idx" class="oo" proc="procprint"> /*Element from the DOM */
  #IDX .data {
    font-size:8pt;
  }

```

In This Example:

To apply the new font size to the first table only, you add the ID for the first section (**#IDX .data**).

Note: You can modify the default ID by using the **ANCHOR=** option in the ODS destination.

CSS Universal Selectors

A *universal selector* is a wildcard that matches every element. You indicate the universal selector by adding an asterisk (*). The example below generates text in Arial font for all of the elements in the document.

```

* {
  font-family:arial;
}

```

ADVANCED CSS SELECTORS

Starting with SAS 9.4, ODS supports many advanced selectors that enable you to target, conceivably, every element. This section discusses some of the advanced selectors such as pseudo-class selectors, attribute selectors, adjacent selectors, and more.

Pseudo-Class Selectors

Pseudo-class selectors enable you to select elements according to their position relative to other elements. These selector names all begin with a colon. Two examples of pseudo-class selectors are **:nth-child(N)** and **:nth-of-type(N)**. These two selectors provide dynamic styling capabilities. The **:nth-child** selector enables you to match an element that is the *n*th child, regardless of its element type. The argument (N) within these two pseudo-class selectors can be a keyword, a number, or an expression of the form *an+b*.

The following sections describe three types of values (keywords, numbers, and expressions) that N can take in a pseudo-class selector.

Keywords

The pseudo-class selector **:nth-child(N)** can take the keyword values where N has a value of **odd** for selecting odd-numbered elements and a value of **even** for even-numbered elements. You can use these keywords as a convenient way to highlight odd and even rows in table, making them easier to read in the output. . For example, the following CSS code modifies the background color for even (red) and odd (green) rows in the table:

```
table tbody tr:nth-child(even) td {
    background-color:red;
}
table tbody tr:nth-child(odd) td {
    background-color:green;
}
```

Numbers

The argument (N) can also take a number value that represents the position for the selected element. For example, the following code uses the **:nth-child** pseudo-class selector to specify a red background for row 5 of a table:

```
table tbody tr:nth-child(5) td {
    background-color:red;
}
```

Expressions

The argument (N) can also be represented as (*an+b*), where *a* and *b* are integers that identify the number of occurrences and where *b* is the first occurrence. For example, the expression **(3n+1)** indicates that the action starts with the first element and then jumps to every third element in the sequence (in this case, 1,4,7,10, and so on). The expression **(2n)** is equivalent to using the keyword **even**, indicating that the code should select every second element. The expression **(4n+2)** matches the second element and then every fourth element (2, 6, 10, and so on).

Additional Pseudo-Class Selectors

There are several other pseudo-class selectors that you can specify, as shown in the list below.

- **:root**
- **:first-child**
- **:first-of-type**
- **:nth-child**
- **:nth-of-type**
- **:empty**
- **:before**
- **:after**
- **:not**

The **:first-child** selector, for example, enables you to select the first element from a parent element.

Note: The list above is not the complete list of available pseudo-class selectors. ODS does not load the entire document into memory. Therefore, certain pseudo-class selectors such as **:last-child** do not appear in the list above.

The following example applies six style options to a table in the Excel ribbon that is duplicated using a CSS with the ODS Excel destination. The six style options that will be applied are shown in the following display in the **Table Style Options** section of an Excel spreadsheet:



Display 1. Modified Table in the Microsoft Excel Style Ribbon

Example Code

```
/* This statement modifies the font for all of the output. */
* th, td {font-family:arial;}

/* This segment targets all th elements within the first */
/* child of the tr element that are descendants of      */
/* thead and table.                                   */
table thead tr:first-child th {
    background-color:royalblue;
    color:white;
    border:2px solid black;
}

/* Targets all td elements within even-numbered tr elements */
/* (rows) that are descendants of tbody and table.          */
table tbody tr:nth-child(even) td {
    background-color:lightblue;
    border:2px solid black;
}

/* Targets the 3rd and 4th th element that are descendants of */
/* tbody and table.                                         */
table tbody th:nth-of-type(3), th:nth-of-type(4) {
    background-color:yellow;
    color:green;
    border:2px double black;
}

/* Targets the td element in the 4th position (column) that */
/* is a descendants of tbody and table.                      */
table tbody td:nth-child(4) {
    background-color:pink;
    border:2px solid black;
}
```

(code continued)

```

/* Targets the first td element (column) and every other td */
/* element thereafter that are descendants of tbody and table. */
table tbody td:nth-child(2n-1){
    background-color:orange;
    border:2px solid black;
}

```

Display 2 shows the output from this example displayed in a mobile device.

Year	Product_Group	Quantity	Profit
1999	A-Team, Kids	286	\$4,980
1999	Bathing Suits, Kids	98	\$1,480
1999	Eclipse, Kid's Clothes	588	\$9,349
1999	Eclipse, Kid's Shoes	334	\$7,137
1999	Lucky Guy, Kids	303	\$7,163
1999	N.D. Gear, Kids	755	\$19,153
1999	Olssons, Kids	208	\$1,975
1999	Orion Kid's Clothes	14	\$289
1999	Osprey, Kids	454	\$7,335
1999	Tracker Kid's Clothes	1243	\$21,848
		4284	\$80,709

Display 2. Applying Elements to a Table with Pseudo-Class Selectors

ATTRIBUTE SELECTORS AND COMBINATORS

With *attribute selectors*, you can target elements based on their attributes and the attribute values. For example, consider the following record from the ODS DOM:

```

<th class="header" type="char" index="1" name="obs">Obs</th>
/* Targets th elements with index attributes. These elements */
/* are descendants of thead and table. */
table thead th[index] {
    background-color:brown;
    font-weight:bold;
    color:beige;
    border:1px solid black;
}

```

(code continued)


```

/* Targets th elements that have the attributes and values */
/* NAME="QUANTITY" and NAME="PROFIT". These elements */
/* are descendants of thead and table. */
table thead th[name="quantity"],
table thead th[name="profit"]
    background-color:green;
    font-weight:bold;
    color:beige;
    border:1px solid black;
}

/* Targets td elements with an attribute and value of */
/* LABEL='YEAR'. These elements are descendants of tbody */
/* and table. */
table tbody td[label="year"] {
    background-color:orange;
}

/* Targets td elements that have an attribute and value of */
/* TYPE="CHAR". These elements are descendants of tbody and */
/* table. */
table tbody td[type="char"] {
    background:beige;
}

/* Targets td elements that have an attribute and value */
/* INDEX="4". These elements are descendants of thead */
/* and table. */
table tbody td[index="4"] {
    background:yellow;
    font-weight:bold;
}

/* Targets th elements with the following attributes and */
/* values: NAME="QUANTITY" and NAME="PROFIT". These elements */
/* are descendants of tbody. */
table tbody th[name="quantity"],
table tbody th[name="profit"] {
    background-color:lightgreen;
    border-top:5pt solid red;
    font-weight:bold;
}

/* This section uses an adjacent selector (td+td) to target td */
/* elements that are not the first element that are descendants */
/* of tbody and table. */
table tbody td+td {
    color:darkblue;
    background-color:lightblue;
}

```

In This Example:

In Output 4, you can see the various attributes that are added based on the color code.

- The NAME= attribute enables you to apply a color to the column name.
- The INDEX= attribute enables you to specify the position or column.
- The TYPE= attribute enables you to add a color based on the data type.

Output 3 shows the output that is created by this example code displayed in the Apple iPad.

Year	Product Group	Quantity	Profit
1999	A-Team, Kids	286	\$4,980
1999	Bathing Suits, Kids	98	\$1,480
1999	Eclipse, Kid's Clothes	588	\$9,349
1999	Eclipse, Kid's Shoes	334	\$7,137
1999	Lucky Guy, Kids	303	\$7,163
1999	N.D. Gear, Kids	755	\$19,153
1999	Olssons, Kids	209	\$1,975
1999	Orion Kid's Clothes	14	\$289
1999	Osprey, Kids	454	\$7,335
1999	Tracker Kid's Clothes	1243	\$21,848
		4284	\$80,709

Output 3. Worksheet That Is Generated from Using Attribute Selectors

There are many other attributes that you can use, but this example is scaled down for demonstration purposes. You can also use various sibling combinators (such as adjacent, general, and sibling selectors) that are based on their positions relative to the other elements in the document tree.

ENHANCING FORMATTED REPORTS IN THE BODY OF AN EMAIL

In the business world, staying relevant also requires that you are one step ahead of what is happening today. For example, some customers might view reports on a desktop machine while others use a mobile device such as an Apple iPhone or iPad. Therefore, you must analyze your customers' needs in order to determine the best medium of delivery for reports. As demonstrated by the examples in the previous section, the ODS Excel destination is flexible enough to enable you to create reports for any medium. In particular, email is a medium that you should consider for a quick and efficient way to review reports because it is efficient for users of both mobile and desktop devices.

You can target delivery of reports through email on mobile devices, desktop computers, and web clients. According to Litmus Analytics, the iPhone leads the market share overall with respect to email users on the mobile platform.¹ Microsoft Outlook is the industry leader when it comes to desktop email, and Google Gmail has the lead share for web email. This section discusses techniques for presenting highly formatted (SMTP) reports across these various email platforms. The discussion centers on including reports in the body of an email message using the under-utilized EMAIL access method in the SAS FILENAME statement.

USING A FILENAME STATEMENT AND A ROUTING METHOD TO INCLUDE FORMATTED OUTPUT IN AN EMAIL MESSAGE

To send formatted output to the body of an email, you need a FILENAME statement that includes the EMAIL access method and a method of routing the formatted output (HTML) to your email device. The routing method can be a DATA step or one of the ODS destinations that are used to generate HTML. The following example shows a basic SAS program that uses the HTML destination to send email.

```
options emailsys=SMTP emailhost=your-mail-host;

filename temp email to="recipient-email-address"
               content_type="text/html"
               from="sender-email-address";
```

(code continued)

¹ Beechler, Drew. 2015. "Top Trends from 2014's Email Client Market Share." Blog. Salesforce Blog. Available at blogs.salesforce.com/company/2015/02/top-trends-from-2014s-email-client-market-share.html. Last viewed March 30, 2015.

```
ods html file=temp rs=none;
ods text="Your text";

proc print data=sashelp.class;
run;

ods html close;
```

In This Example:

- To send formatted output to the body of an email, you must use the Simple Method Transfer Protocol (SMTP). You set that protocol using the EMAILSYS=SMTP option in the OPTIONS statement, as shown in the example above.
- The EMAILHOST= system option defines your mail host.
- The TO= option (with addresses for the email recipients) is the only required option in the FILENAME statement.
- The ODS HTML destination and the PRINT procedure are used to route and print the formatted output. In the ODS HTML destination, using the RS=NONE option forces the output to be written as record-based data rather than a single stream of data.
- The ODS TEXT= option displays the text.

Note: The FILENAME statement has some other optional options you can use, too (for example, you can add a subject to your email with the SUBJECT= option). However, the CONTENT_TYPE= option is the most important of the optional options. This option specifies the Multipurpose Internet Mail Extensions (MIME) that are used to send email. The MIME type indicates to the email client what format of email to expect. In Microsoft Windows operating environments, the default format is simple text (plain text). But with simple text, you only see the HTML markup rather than formatted output. For complete information about FILENAME statement options, see the appropriate Base SAS® documentation for your release of SAS.

GENERATING FORMATTED REPORTS IN EMAIL: LIMITATIONS

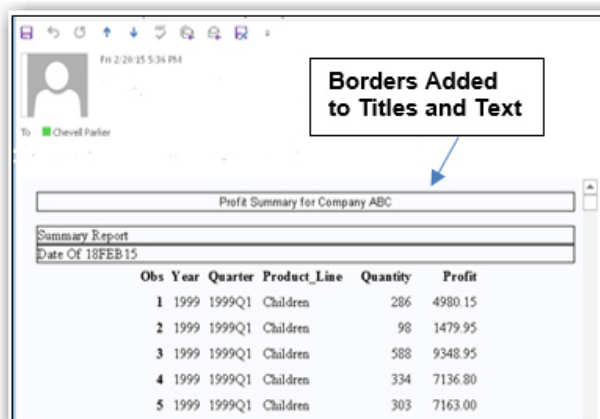
When you send a formatted report to the body of an email, there are a number of limitations of which you need to be aware:

- One of the greatest limitations is size of the file you can send. Many mail servers have limitations regarding files size. You also need to be aware of file size when it comes to email that will be opened on mobile devices because file size can affect the amount of time it takes to load an email message.
- With regard to displaying email, the email client ignores JavaScript because it has the potential to be harmful. The client might also consider the following HTML tags to be unimportant in terms of displaying the mail: `<script>`, `<iframe>`, `<object>`, `<embed>`, `<applet>`, and `<param>`. Some email clients only have sparse support for CSS. For example, some clients might permit a CSS only inline. This behavior is true for clients such as Google Gmail, which removes all embedded CSS information between the `<head>` and `</head>` tags. In addition, versions of IBM Notes (formerly Lotus Notes) before version 9 do not have good support for CSS.
- Positioning and inclusion of background images with CSS are supported only in a few email clients.
- HTML accessibility is not largely supported on email clients.
- For security reasons, scripting is not supported.
- You cannot animate images in the formatted content.
- Flash objects are not supported.
- Support for graphics varies, depending on the client.

GENERATING FORMATTED REPORTS IN EMAIL: CHALLENGES

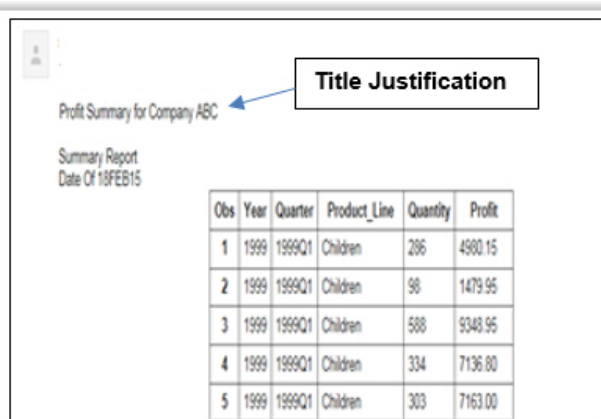
The ability to render graphics easily in the body of the email is a challenge for many email clients. For example, Microsoft Outlook has one of the larger email user bases and is also one of the more challenging email clients to target because of its rendering engine. As a result, you should always verify the display of emails within that client. Beginning with Outlook 2007, Microsoft switched its email rendering engine from Internet Explorer 6 to Microsoft Word. This switch is what makes Outlook more challenging. The switch created a number of issues because email messages no longer look the same as they did in previous releases of Outlook. The appearance issues occur because Word's support for HTML is not as robust as that of Internet Explorer. However, Outlook on the Apple Macintosh does not have this problem because it uses the WebKit rendering engine. Other email clients have had similar challenges because of their level of support for CSS.

Consider [Display 2](#), which is the result of email that was sent using the EMAIL access method and ODS HTML in SAS 9.4. If you compare this display with [Output 4](#), you can see that the email in Display does not contain certain style information for features. For example, it does not have a background color for the headers. However, Display 2 does place borders around the title, footnotes, and text strings that are generated with an ODS TEXT= statement. But these borders should not appear in this email because they do not appear that way in the browser. On the other hand, Display 3 is an example of Gmail for which the embedded style is removed completely, which causes a lack of style information and justification. (See SAS Note [55410](#), "Borders are displayed around titles and footnotes in ODS HTML output that is viewed within Microsoft Outlook," for instructions about how to remove the border from the HTML destination.)



Obs	Year	Quarter	Product_Line	Quantity	Profit
1	1999	1999Q1	Children	286	4980.15
2	1999	1999Q1	Children	98	1479.95
3	1999	1999Q1	Children	588	9348.95
4	1999	1999Q1	Children	334	7136.80
5	1999	1999Q1	Children	303	7163.00

Display 2. Formatted Email in Microsoft Outlook



Obs	Year	Quarter	Product_Line	Quantity	Profit
1	1999	1999Q1	Children	286	4980.15
2	1999	1999Q1	Children	98	1479.95
3	1999	1999Q1	Children	588	9348.95
4	1999	1999Q1	Children	334	7136.80
5	1999	1999Q1	Children	303	7163.00

Display 3. Formatted Email in Google Gmail

USING ODS DESTINATIONS TO GENERATE FORMATTED EMAIL

When you generate formatted email using the ODS HTML destination, an embedded CSS file is added by default. Even though such output might look good in your browser, Outlook, for example, does not handle the HTML file well with the way in which styles are applied. Viewing email that is generated with Microsoft Outlook causes problems similar to those shown in [Display 2](#) above.

However, ODS has three destinations you can use to produce HTML that is better suited for rendering formatted email across email clients:

- HTML3: This destination generates HTML 3.2 markup, which does not use an embedded CSS. It only uses markup tags. This destination is a good option for generating email across all email clients since some of the older clients do not handle CSS well.
- MSOffice2K: The MSOffice2K destination generates HTML that is optimized for Microsoft Office applications.

(list continued)

- TableEditor tagset: Similar to the MSOffice2K destination, the TableEditor tagset also uses HTML that is optimized for Office products that use the inline CSS. The TableEditor tagset also has default meta tags that enable a responsive design for mobile clients. You can download the tagset from the "Introduction" section of the web page "Base SAS: Creating a Data Grid Like VB.NET." (support.sas.com/rnd/base/ods/odsmarkup/tableeditor/index.html)

The following program uses all three of the ODS destinations to render output:

```
Filename output_email to="john.doe@gmail.com"
                        content_type="text.html"
                        subject="Profit Report";

*ods html3 file=output style=HTMLblue;
ods msoffice2k file=output style=HTMLblue
               metatext='name="viewport" content="width=device-width" '
               options(pagebreak="no");
*ods tagsets.Tableeditor file=output options(format_email='yes'
                                             pagebreak="no")
               style=HTMLblue;

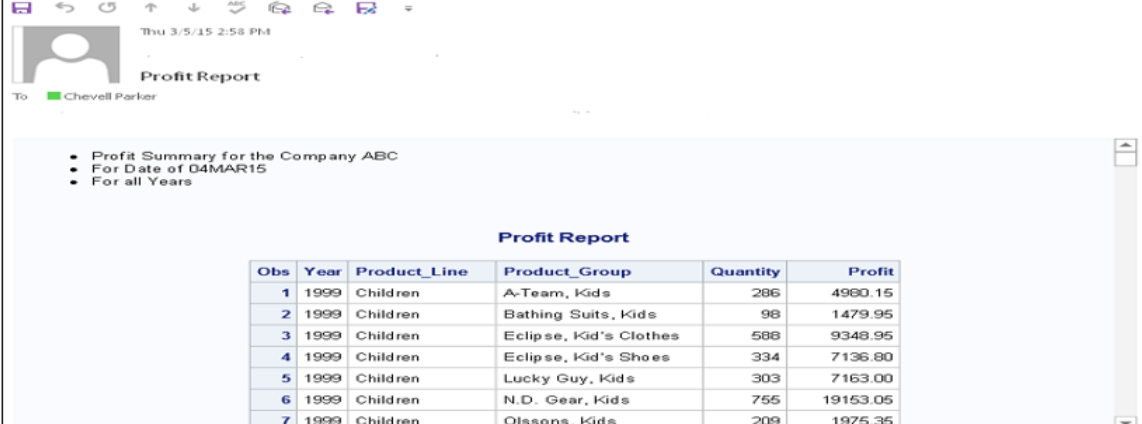
title1 "Profit Report";
proc odstext;
  list;
    item "Profit Summary for the Company ABC";
    item "For Date of &sysdate";
    item "For All Years";
  end;
run;

proc print data=sashelp.orsales;
  var Year Product_Line Product_Group Quantity Profit;
run;
ods _all_ close;
```

In This Example:

- This code shows three ODS destinations that work best when you are creating formatted output for email. **Note:** Two of the destinations appear in the code (ODS HTML3 and ODS TAGSETS.TABLEEDITOR), but they are commented out because the output for this example is only shown for the ODS MSOFFICE2K destination.
- The MSOffice2K destination adds the VIEWPORT metatag, which sets the output width of the mobile devices to the device width. The width that is specified can be the actual width of the device, or you can use WIDTH="device-width", which dynamically obtains the width of the device.
- The ODS TEXT= statement incorrectly generates borders around text strings in Outlook. To avoid that problem, this example uses PROC ODSTEXT to add text in the output.

Output 4 show the output that is generated by the HTML3 destination, the MSOffice2K destination, and the TableEditor tagset.



Thu 3/5/15 2:58 PM

Profit Report

To: Chevell Parker

- Profit Summary for the Company ABC
- For Date of 04MAR15
- For all Years

Profit Report

Obs	Year	Product_Line	Product_Group	Quantity	Profit
1	1999	Children	A-Team, Kids	286	4990.15
2	1999	Children	Bathing Suits, Kids	98	1479.95
3	1999	Children	Eclipse, Kid's Clothes	588	9348.95
4	1999	Children	Eclipse, Kid's Shoes	334	7136.80
5	1999	Children	Lucky Guy, Kids	303	7163.00
6	1999	Children	N.D. Gear, Kids	755	19153.05
7	1999	Children	Olssons, Kids	209	1975.35

Output 4. Output That Is Generated with the HTML3 Destination, the MSOffice2K Destination, and the TableEditor Tagset

USING GRAPHICS WITHIN EMAIL

If you generate a formatted email, it is likely that you will want to include a logo or image on the report. You can reference an external image that is stored on a web server by using a URL (HTTP) addressing method so that everyone has access to it. This method works on a majority of email clients as long as the user is connected to the Internet. However, a common mistake that people make is to reference an image that is on their local machines. The HTML5 destination in SAS 9.4 provides the ability to embed scalable vector graphics within the HTML file that is generated. However, support for this capability is minimal across email clients.

Now, you can embed an image directly in your report by using the SAS INLINED= suboption, which is new in the second maintenance release for SAS 9.4 (TS1M2). This option enables you to specify a reference name that you can use to embed an attachment in the body of an email message. The INLINEDs option is one of the attachment suboptions you can use in the FILENAME statement's ATTACH= option. This suboption helps you to create a multipart email message: one part is text and HTML; the other part is an image file (for example, a GIF file) that is transferred using base 64 encoding. With this method, the image is attached, but it is hidden. You can display the image the email by using a content ID. For example, if your reference name is `logo`, the HTML image tag and content ID is ``.

The following example embeds a SAS logo into email that is displayed in Outlook:

```
options emailsys=smtplib emailhost=your-email-host;
filename output email to="jane.doe@gmail.com"
                    sender="joe.smith@gmail.com"
                    attach=('C:\SAS.jpg' inlined="logo")
                    subject="Report for Company XYZ"
                    content_type="text/html";

ods msoffice2k file=output rs=none;
title j=1 '<img src=cid:logo height="100" width="120">';
title2 "Profit Report for Company XYZ";

proc print data=sashelp.orsales;
run;

ods msoffice2k close;
```

In This Example:

- The image is embedded by using the INLINED= suboption inside the ATTACH= option. The INLINED= reference in this case is logo.
- The content ID (CID) references logo in the tag that is in the first TITLE statement. This statement inserts text into your output.

Output 5 shows the result of this program in Outlook email:



Output 5. A SAS Logo Embedded in Microsoft Outlook Email

ENHANCING AND RENDERING REPORTS WITH THE HTML AND HTML5 ODS DESTINATIONS

So far in this discussion, you have seen how important HTML is in delivering formatted email to email clients. This section explains how to take your reporting to the next level by taking advantage of some new features that are available to the ODS HTML destinations. In addition, this section discusses how you can use the new level 3 CSS (or CSS3) to enhance the actions of the ODS HTML5 destination and how you can add video and audio to a file using the ODS HTML5 destination.

This section also discusses how to use the new STREAM procedure in the process of your web development. Finally, you will learn about the ODS layout feature and how you can use it to your advantage with the ODS HTML and ODS HTML5 destinations.

THE BENEFITS OF USING THE NEW ODS HTML5 DESTINATION

For good reason, the software industry is turning to HTML5 as its choice for presenting web content rather than with older technologies such as Adobe Flash and Microsoft Silverlight. HTML5 output has better cross-browser capabilities (for example, sharing content across desktop computers, mobile devices, and other devices that support HTML5). In addition, many third-party applications (such as Flash and Silverlight) that were previously used for document setting are being replaced with CSS3, which works in tandem with HTML5. You now have the ability to animate web pages using strictly CSS. This same task required large amounts of JavaScript in the past.

Given the benefits of HTML5 and its use industry wide, the ODS HTML5 destination is a logical choice for your toolkit. This destination is new in SAS 9.4, and continuous enhancements to the destination are expected in future releases. This destination boasts a number of powerful features:

- a more robust tagging structure
- the ability to embed video and audio
- drag-and-drop support
- editable content
- the ability to create graphics with JavaScript and the HTML5 canvas method

Once you create HTML output, you can move that output to mobile devices. However, for mobile devices, you might need to make some adjustments. For example, if you have long column headings, you can conserve space by rotating those headings. The TRANSFORM property enables you to perform many actions such as rotating, scaling, moving, and skewing objects. In this example, the TRANSFORM property in the STYLE= suboption rotates headers 20 degrees. In addition, this example adds a video clip to the report.

```
proc template;
  define style styles.test;
    parent=styles.htmlblue;
    class header /
      prehtml="<div style='transform:rotate(20deg)'">"
      posthtml="</div>";
    end;
run;

ods html5 file="c:\temp.html" style=styles.test;

proc print data=sashelp.shoes(obs=5)
  style=(header obsheader)={height=25 vjust=middle};
title "Sales Report for the Quarter";
title2 "See the following video for more detail about the latest quarter.";
run;

data _null_;
dcl odsout obj();
obj.video(file: "c:\temp\IMG_0359.MOV",
  width: "300",
  height: "300",
  preload: "auto",
  autoplay: "no",
  poster: "c:\quarterly.jpg",
  muted: "yes");
run;
ods html5 close;
```


Output 6 shows the report modified for mobile devices with the video attached.

Sales Report for the Quarter

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	\$29,781	\$191,821	\$769
2	Africa	Men's Casual	Addis Ababa	4	\$67,242	\$118,036	\$2,284
3	Africa	Men's Dress	Addis Ababa	7	\$76,793	\$136,273	\$2,433
4	Africa	Sandal	Addis Ababa	10	\$62,819	\$204,284	\$1,861
5	Africa	Slipper	Addis Ababa	14	\$68,641	\$270,795	\$1,771

See the following video for more detail about the latest quarter.

Output 6. Using the HTML5 Destination to Display Rotated Headers and to Include a Video

USING PROC STREAM WITH THE ODS HTML AND ODS HTML5 DESTINATIONS

PROC STREAM is a new procedure in SAS 9.4, and it was pre-production in SAS 9.3. This procedure enables you to process an input stream that consists of arbitrary text that can contain SAS macro specifications. Within this input stream, macro specifications are expanded while the text, which includes SAS statements, is simply passed to the stream as invalidated text. As a result, the SAS parser does not attempt to execute the SAS statements.

You can use PROC STREAM as a stand-alone procedure to generate custom and dynamic web pages. You can also use it with ODS to add dynamic headers and footers or even to add dynamic content (such as check boxes or selecting lists) from the data. In addition, you can generate SAS server pages with the procedure or embed macro variables within HTML files that are included, making the possibilities of its use even more powerful.

USING ODS WITH PROC STREAM

As mentioned previously, PROC STREAM can generate custom and dynamic HTML pages either alone or in conjunction with ODS. Consider the following example in which an initial macro is created. This macro uses the ODS HTML destination along with the NOTOP and NOBOT options to generate a web page that does not have the opening and closing parts of the file. That is, the file only consists of tables, graphs, and text.

```
filename temp "c:\temp.html";

%macro create_table;
  ods html file=temp(nobot notop);

  proc print data=sashelp.orsales(obs=5);
    title;
    run;
  ods html close;
%mend;
```

(code continued)

```

filename output "c:\stream.html";
proc stream outfile=output prescol resetdelim='label';
  begin
    <html>
      <head>
        <style>
          .header {background-color:lightblue}
          .footer {font-weight:bold}
          h2,h3 {text-align:center}
        </style>

        <h2>Report for Company ABC</h2>
        <h3> Date: %sysfunc(date(),date.); </h3>
      </head>

      <body>
        label;
        %let rc=%sysfunc(dosubl('%create_table'));
        %include temp;
        %let rc=%sysfunc(dosubl(
          proc sql noprint;
            select sum(profit) %str(,) avg(profit) %str(,) max(profit)
              into:sum %str(,):avg %str(,):max
              from sashelp.orsales;

          quit;
        ));

        <table align=center frame=box>
          <tr>
            <td class="footer">Sum_Profit:
              %sysfunc(putn(&sum,dollar14.2))</td>
          </tr>
          <tr>
            <td class="footer">Avg_Profit:
              %sysfunc(putn(&avg,dollar10.2))</td>
          </tr>
          <tr>
            <td class="footer">Max_Profit:
              %sysfunc(putn(&max,dollar10.2))</td>
          </tr>
        </table>
      </body>
    </html>
  ;;;;

```

In This Example:

- The style information in PROC STREAM is simply added as text.
- The initial <h2> tag is simply passed as text and is not validated. That is, the tag is not sent to the SAS processor.

(list continued)

- The secondary header uses the SAS DATE() function within the %SYSFUNC macro function. The DATE() function returns the current date. The macro function is executed because it is a macro trigger.
- In order for the SAS syntax statements in the %CREATE_TABLE macro to be executed, you must place the macro within the SAS DOSUBL function that is within the %SYSFUNC macro function. The DOSUBL function prevents the SAS statements from being passed as text.
- The %SYSFUNC macro function and the DOSUBL function are used later in the program to execute a PROC SQL step. The PROC SQL step uses the INTO clause from the summary functions SUM, AVG, and MAX of the column profit to generate three macro variables. In this example, PROC STREAM create a SAS server page by dynamically building summaries for the HTML table that is within the procedure.
- The stream is ended with four semicolons.

Output 7 shows the output that is generated by this program.

Report for Company ABC								
Date: 04MAR15								
Obs	Year	Quarter	Product_Line	Product_Category	Product_Group	Quantity	Profit	Total_Retail_Price
1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40
3	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Clothes	588	9348.95	18768.80
4	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Shoes	334	7136.80	14337.20
5	1999	1999Q1	Children	Children Sports	Lucky Guy, Kids	303	7163.00	12996.20
Sum_Profit: \$59,085,048.00 Avg_Profit: \$64,786.24 Max_Profit: \$552,970.50								

Output 7. Appending a Header and a Footer to Output Using the STREAM Procedure

USING THE ODS LAYOUT FACILITY WITH THE ODS HTML AND ODS HTML5 DESTINATIONS

The ODS layout feature became production with SAS 9.4, although a pre-production version is available in earlier releases. This feature enables you to panel tables, graphics, and text much like the GREPLAY procedure does for graphics and the HTMLPANEL tagset does for both tables and graphics. ODS layout creates two different types of layouts: absolute layouts and gridded layouts. With the *absolute layout*, you can specify X and Y coordinates that enable you to pinpoint exactly where you want to place output on a page. With *gridded layout*, on the other hand, you can panel rows and columns. Both types of layouts consist of regions. However, unlike the absolute layout, gridded layouts dynamically size the regions to be large enough to accommodate the output. Gridded layout also enables the output to span multiple pages, if necessary. The ODS HTML and HTML5 destinations only use the gridded layout.

Some users prefer to use the HTMLPANEL tagset or the TableEditor tagset, both of which work well for the features they contains. However, an entire infrastructure has been built to accommodate ODS layout. This infrastructure is more flexible than tagsets because it includes new options, functionality, and methods for modifying a layout using templates. In addition, you can use ODS layout with other destinations. That is, there are options to control every facet of layout, and you have the ability to control the style of the layouts from the template. The paneling abilities in ODS layout are flexible and powerful, so you should consider using it over the HTMLPANEL tagset to handle reporting needs that require you to combine or panel tables or graphics on a page. For example, ODS layout offers the options COLUMN_GUTTER= and ROW_GUTTER= to add more space between tables or graphs within a panel. This functionality is not possible with the tagsets. Be aware that ODS layout is not supported for the ODS Excel nor the ODS RTF destinations. However, it works extremely well with the ODS HTML, ODS PDF, and ODS POWERPOINT destinations.

The following example illustrates how to use ODS layout to create a three-column layout in output, based on the BY group:

```
proc sort data=sashelp.class out=temp;
    by age;
run;
ods html file='temp.html';
title "Gridded Layout That Creates Three Columns Based on a BY Group";

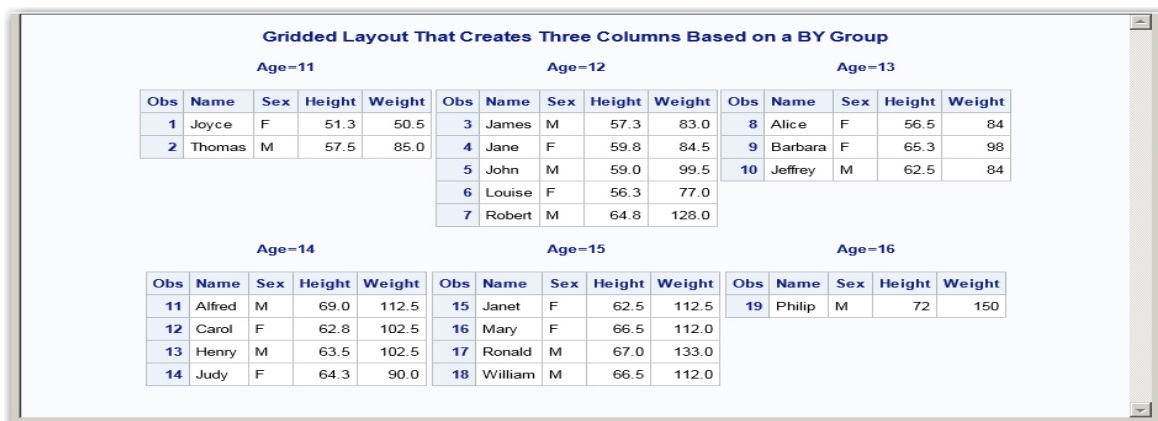
ods layout gridded advance=bygroup columns=3 column_gutter=0;
ods region;
proc print data=temp;
    by age;
run;

ods layout end;
ods html close;
```

In This Example:

- The layout in this example is accomplished by using the COLUMNS=3 and the ADVANCE=BYGROUP options in the ODS LAYOUT statement.
- The COLUMN_GUTTER=0 option generates no space between the tables.
- The output is then generated with the ODS HTML destination.

Output 8 shows the resulting gridded-layout output.



Age=11					Age=12					Age=13				
Obs	Name	Sex	Height	Weight	Obs	Name	Sex	Height	Weight	Obs	Name	Sex	Height	Weight
1	Joyce	F	51.3	50.5	3	James	M	57.3	83.0	8	Alice	F	56.5	84
2	Thomas	M	57.5	85.0	4	Jane	F	59.8	84.5	9	Barbara	F	65.3	98
					5	John	M	59.0	99.5	10	Jeffrey	M	62.5	84
					6	Louise	F	56.3	77.0					
					7	Robert	M	64.8	128.0					
Age=14					Age=15					Age=16				
Obs	Name	Sex	Height	Weight	Obs	Name	Sex	Height	Weight	Obs	Name	Sex	Height	Weight
11	Alfred	M	69.0	112.5	15	Janet	F	62.5	112.5	19	Philip	M	72	150
12	Carol	F	62.8	102.5	16	Mary	F	66.5	112.0					
13	Henry	M	63.5	102.5	17	Ronald	M	67.0	133.0					
14	Judy	F	64.3	90.0	18	William	M	66.5	112.0					

Output 8. Using Gridded Layout to Add Three Columns Based on BY Group

USING ODS DESTINATIONS AND FEATURES WITH SAS® 9.4 IN PRACTICAL APPLICATIONS

The Excel destination is pre-production beginning in the first maintenance releases for SAS 9.4 (TS1M1), and it is expected to become a production feature in SAS 9.4 TS1M3. This ODS destination generates output in the native 2010 Microsoft Excel format, which is the XML Open Office format. The Excel destination also includes new features such as CSS advanced techniques and the use of two new SAS procedures, ODSTEXT and ODSLST. Using the Excel destination and these new capabilities, you can now take full advantage of the ODS Report Writing Interface (RWI). As described in the section "The Report Writing Interface" of "Chapter 1: Introduction" in *Getting Started with the SAS® 9.4 Delivery System*, this interface enables you to "create highly customized reports in an object-oriented language that is fully integrated with ODS." The RWI has a number of methods (the most common method is the table method) that enables you to create highly customized output.

GENERATING A CUSTOMIZED TABLE OF CONTENTS USING THE EXCEL DESTINATION

The ODS Excel destination makes it easy for you to create a customized table of contents (TOC). With the CONTENTS= option, you can generate a default TOC. However, there are times when you want to customize the TOC more than the CONTENTS= option allows. There are methods for modifying the text that is displayed and the style. But suppose that you want to modify the style for each entry that you add or that you want to have multiple entries that cover a single table. Or, you might want to modify where the user lands on the sheet to which you are linking. You can do all of these things by using the Excel destination. You can also add free text in a customized TOC.

The next example uses the ODS TEXT procedure and the Excel destination to generate a combination TOC that contains a list and arbitrary text.

```
ods excel file="c:\temp20.xlsx" options(embedded_titles="yes"
                                         sheet_name="Table of Contents");

proc odstext;
  p 'Table of Contents by Region' / style={font_weight=bold
                                           font_size=12pt};

  list;
    item "Detail for Region Africa" /
      style={url="#Africa!A1" color=blue tagattr="mergeacross:4"
            };
    item "Summary for Region Africa" / style={url="#Africa!E62"
                                           color=red};

    item "Detail for Region Asia" /
      style={url="#Asia!A1" color=blue};
    item "Summary for Region Asia" / style={url="#Asia!E20" color=red
                                           };

    item "Detail for Region Canada" /
      style={url="#Canada!A1" color=purple};
    item "Summary for Region Canada" / style={url="#Canada!E43"
                                           color=red};

    item "Detail for Region Central America-Caribbean" /
      style={url="#'Central America-Caribbean'!A1" color=blue};
    item "Summary for Region Central America-Caribbean" /
      style={url="#'Central America-Caribbean'!E36" color=red};
    . . .more ITEM statements. . .
  end;

  p ' ';
  p 'The Table of Contents information links to both the detail records
    and the summary of sales for this region.';
  p ' ';
  p '* Items that are listed in purple were not profitable over the last
    fiscal year.' / style=systemfooter[color=purple];
run;

ods excel options(sheet_name="#byval(region)" sheet_interval="bygroups");
proc sort data=sashelp.shoes out=regions;
  by region;
run;

proc print data=regions;
  by region;
  sum sales;
run;

ods excel close;
```

In This Example:

- PROC ODSTEXT uses the ITEM and P statements to accomplish the final product for this example.
- The URL= attribute targets the various BY groups and summaries that determine where the cursor lands on the worksheet. The cursor's position is referenced with the worksheet name and cell reference (*#worksheet-name!cell-reference*).
- Styles are applied to various pieces of text by using the STYLE= option in the P statements. The MERGEACROSS attribute is added in the TAGATTR= attribute for the first table. This attribute enables the entire text that is specified to become a link.

Output 9 shows the cover page that is generated by the example code.



Output 9. Creating a Custom Table of Contents with the ODS ODSTEXT Procedure and the Excel Destination

USING THE ODS EXCEL DESTINATION WITH THE REPORT WRITING INTERFACE

The Report Writing Interface (RWI), which is fully supported in the Excel destination, is an advanced ODS technique that gives you control and flexibility in creating reports. The RWI, formerly known as the object-oriented DATA step, provides a number of powerful capabilities:

- You can specify methods for building reports by using object dot syntax in a DATA step.
- You can perform tasks that are traditionally difficult in ODS (for example, merging columns and rows in a report).
- Many methods are available through the RWI. However, the Layout and Image methods currently are not supported in Excel.

The following example is a basic demonstration of the power of the RWI within the ODS Excel destination. This example merges two column headings and dynamically creates or computes new columns.

```
data one;
  input market $10. +1 value comma10.2 change;
  cards;
S&P500      2,025.00 200
DOW         17,526.00 300
NASDAQ      4165.50 200
US_Dollar   92,751.00 500
Crude_Oil    47.52   -3
T-Notes     129,000.00 -5000
;
run;
```

(code continued)

```

ods escapechar="^";
ods excel file="c:\temp.xlsx" options(embedded_titles="yes");
data _null_;
  set one end=done;
  if _N_ = 1 then do;
    declare odsout obj();
    obj.title(data:"%sysfunc(today()),worddate.");
    obj.table_start();
    obj.row_start();
    obj.format_cell(data:"Morning Markets",column_span:4, style_attr:
      "background=red color=yellow");
    obj.row_end();
  end;
  percent=(value/change)/100;
  obj.row_start();
  if change > 0 then
    obj.format_cell(data: market,style_attr:"pretext=
      '^{\style[color=green] ^{\unicode 2B06}}' background=#CCFFFF just=1");
  else obj.format_cell(data:market,style_attr: "pretext=
    ^{\style[color=red] ^{\unicode 2B07}}' background=#CCFFFF just=1 ");
    obj.format_cell(data: value, style_attr:" background=#CCFFFF ");
    obj.format_cell(data: change,style_attr:" background=#CCFFFF ");
    obj.format_cell(data: percent,style_attr:" background=#CCFFFF",
      format:"percent10.2");
    obj.row_end();
  if done then do;
    obj.row_start();
    obj.format_cell(data:"Data as of %sysfunc(time()),time.",
      column_span: 4,style_attr: "backgroundcolor=red
      color=yellow");
    obj.row_end();
    obj.table_end();
  end;
run;

ods excel close

```

In This Example:

- The example starts by checking the value of the `_N_` variable and instantiating the object OBJ by using the `DECLARE` statement. It also adds the `TABLE.START()`, `ROW_START()`, and `FORMAT_CELL()` methods. The `FORMAT_CELL()` method uses the `COLUMN_SPAN` option to add the spanning header, or merged text, over four cells. The `ROW_END()` method is used then to end the row.
- The `PRETEXT=` attribute creates up and down arrows, based on the value of the `CHANGE` variable, in the left column.
- The program uses the variable `DONE` (in the `SET` statement) to verify that it is on the last record. Then the program uses the `FORMAT_CELL()` method and the `COLUMN_SPAN:4` option to merge and add text. The `ROW_END()` method is used again to close the row.
- The `TABLE_END()` method closes the entire table.

Output 10 shows the result of this program.

January 20, 2015				
Morning Markets				
↑S&P500	2025	200	10.13%	
↑DOW	17526	300	58.42%	
↑NASDAQ	4165.5	200	20.83%	
↑US_Dollar	92751	5000	18.55%	
↓Crude_Oil	47.52	-3	(15.84%)	
↓T-Notes	129000	-5000	(25.80%)	
Data as of 22:38:18				

Output 10. Generating Output with the Report Writing Interface and the Excel Destination

ADDING PRINTED PAGE BREAKS IN THE OUTPUT OF EXCEL OUTPUT

Starting in the third maintenance release for SAS 9.4 (TS1M3), you can use the Excel destination to add printed page breaks to a report. You can add a page break after each object, output, procedure, or BY group using the ODS Excel option ROWBREAKS_INTERVAL=PROC | OUTPUT. With this capability, you now can place tables in the same worksheet and request printed page breaks at the end of the tables. This feature is very useful for BY groups when you need to distribute the output based on the grouping.

The following example the ROWBREAKS_INTERVAL= option is set to OUTPUT. The SHEET_INTERVAL= option is included also in order to add both worksheets to the current sheet.

```
proc sort data=sashelp.class out=test;
  by sex;
run;

ods excel file="c:\temp.xlsx" options(rowbreaks_interval="output"
                                     sheet_interval="none");

proc print data=test;
  by sex;
run;

ods excel close;
```

Output 11 illustrates the use of the ROWBREAKS_INTERVAL option to add a page break between sections of output.

Sex=F				
Obs	Name	Age	Height	Weight
1	Alice	13	56.5	84.0
2	Barbara	13	65.3	98.0
3	Carol	14	62.8	102.5
4	Jane	12	59.8	84.5
5	Janet	15	62.5	112.5
6	Joyce	11	51.3	50.5
7	Judy	14	64.3	90.0
8	Louise	12	56.3	77.0
9	Mary	15	66.5	112.0

Sex=M				
Obs	Name	Age	Height	Weight
10	Alfred	14	69.0	112.5
11	Henry	14	63.5	102.5
12	James	12	57.3	83.0
13	Jeffrey	13	62.5	84.0
14	John	12	59.0	99.5
15	Philip	16	72.0	150.0
16	Robert	12	64.8	128.0
17	Ronald	15	67.0	133.0
18	Thomas	11	57.5	85.0
19	William	15	66.5	112.0

Output 11. Using the ROWBREAKS_INTERVAL Option to Generate Page Breaks

ADDING NATIVE EXCEL GRAPHICS TO YOUR EXCEL WORKSHEETS

Beginning in the production release of SAS 9.4 (TS1M3), it is planned for you to be able to generate native Excel graphs using the new MSCHART procedure (the procedure will be a pre-production feature in that release). If you add PROC MSCHART within the ODS Excel destination, you can generate true native Excel graphics rather than having to embed an image. This aspect distinguishes this procedure from other procedures (for example, SG or SAS/GRAPH® procedures) that enable you to use graphics.

In the following example, PROC MSCHART is used to tie the graphic to the table. Therefore, any change in the table data also updates the graph. This action enables you to use the interactive capabilities of the native graph to do such things as add styles, filter, and modify chart elements. If you use this outside of ODS Excel, a simple image is generated. Other statements that are available in the third maintenance release of SAS 9.4 (TS1M3) are HCOLUMN, VCOLUMN, LINE, PIE, and SCATTER.

```
/* Summarize the data. */
proc sql;
  create table summary as
    select(region), sum(sales) format=dollar14.2 as sales
    from sashelp.shoes
    group by region;

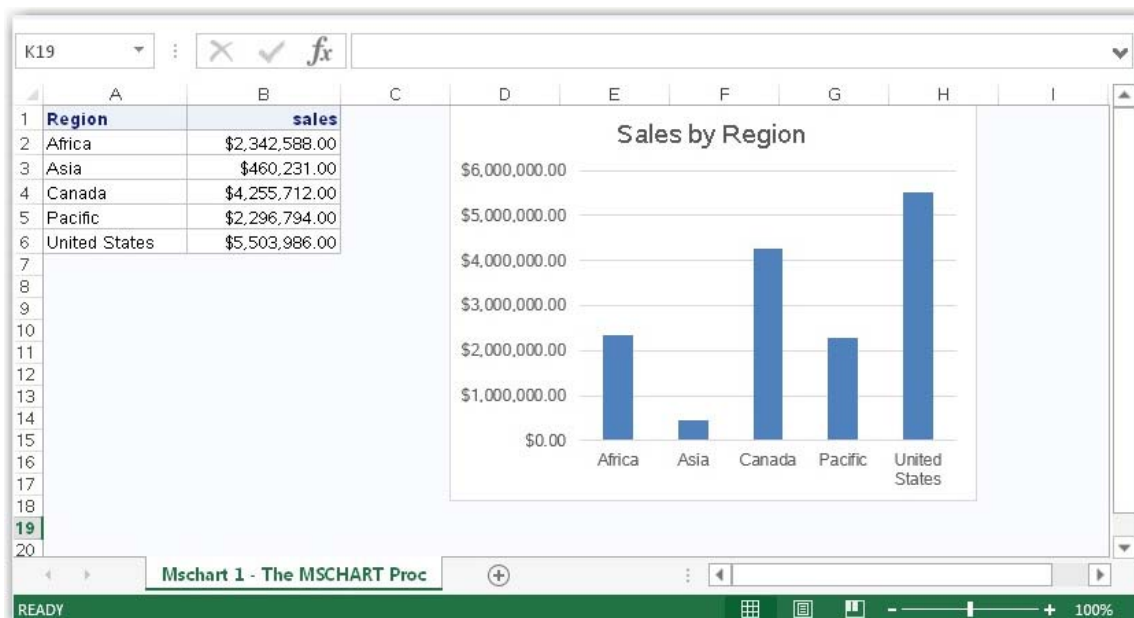
run;
quit;

ods excel file="c:\temp.xlsx";

title "Sales by Region";
proc mschart data=work.summary category=region width=4in position="$D$1";
  where region in("Africa","Asia","Canada","Pacific","United States");
  vcolumn sales;
run;

ods excel close;
```

Output 12 shows the final result of this program.



Output 12. Generating a Native Microsoft Excel Graph Using the MSCHART Procedure

CONCLUSION

Staying relevant in the business world is easy when you know about and use the right tools, and the SAS Output Delivery System has plenty of tools from which to choose! With these tools, you can take advantage of so many features that range from the capabilities of CSS style engine and the strengths of HTML or HTML5 to the ability to generate powerful reports and outstanding output in email with the new SAS 9.4 ODS destinations. In keeping with the changing nature of technology, these tools provide the flexibility for you to offer your reports on any device that your users prefer.

REFERENCE

Smith, Kevin D. 2013. "Cascading Style Sheets: Breaking Out of the Box of ODS Styles." Proceedings of the SAS Global Forum 2013 Conference. Cary, NC: SAS Institute Inc. Available at support.sas.com/resources/papers/proceedings13/365-2013.pdf.

RECOMMENDED READING

Bos, Bert. 2015. W3C: Cascading Style Sheets. Available at www.w3.org/Style/CSS/Overview.html. Accessed on March 16, 2015.

Smith, Kevin D. 2011. "Unveiling the Power of Cascading Style Sheets (CSS) in ODS." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc. Available at support.sas.com/resources/papers/proceedings11/297-2011.pdf.

ACKNOWLEDGEMENTS

I would like to thank Susan Berry for her help on this paper.

CONTACT INFORMATION

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

Chevell Parker
SAS Institute Inc.
Cary, NC
Email: support@sas.com
Web: support.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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