

The Power to Show: Ad Hoc Reporting, Custom Invoices, and Form Letters

Daniel O'Connor, SAS Institute Inc.

ABSTRACT

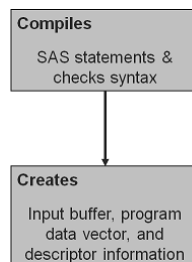
The ability to prepare complex reports and convey your message in a clear and concise manner is an absolute imperative in today's sophisticated business environment. Clearly designed reports can enhance your organization's credibility and reputation. DATA_NULL_ report writing has long been an integral part of the custom report writing offered by SAS®, but with this newly updated ODS Report Writing technology in SAS® 9.2, you will have the ability to produce reports that you have only dreamed about. These new features will allow you to build custom data-centric reports in an easy-to-use object-oriented manner that is fully integrated with the ODS System. This technology is perfectly suited for creating custom invoices, inserting narrative descriptions in a table or document, creating form letters and non-rectangular reports, inserting custom subtotals, and it will address a variety of custom reporting needs.

INTRODUCTION

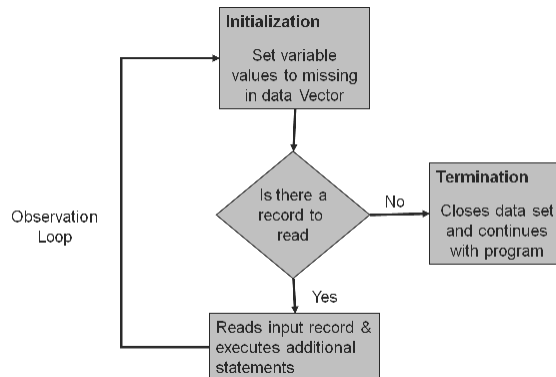
The DATA step is an essential concept in SAS. The DATA step serves many purposes like creating SAS data sets, custom report writing, file management, and information retrieval. The ODS Report Writing Interface will exclusively focus on the custom report writing capabilities. The custom report writing capabilities are also more commonly referred to as DATA_NULL_ report writing which has long been an integral part of the SAS reporting solution. The ODS Report Writing Interface is intended to fully embrace ODS features such as proportional fonts, traffic lighting, using colors, images, Unicode characters, while at the same time providing pixel perfect placement capabilities. This interface is not only fully integrated with all the capabilities of the ODS System, but also takes advantage of the rich programming features that the DATA step offers such as conditional logic, formatting capabilities, by-group processing, arrays, and a wealth of other features. The ODS Report Writing Interface is an object-oriented language that provides you with flexibility and control so that even the most rigid reporting requirements can be met with ease.

DATA STEP PROCESSING

Data step processing has two distinct phases: the compilation phase and the execution phase. The compilation phase checks for syntax of the SAS statements and then compiles them creating the input buffer, program data vector, and descriptor information.



The execution phase is executed once for each input observation that is being processed. The execution phase is further broken down into three categories: the initialization, observation loop, and the termination processing. These three phases are particularly important to the report writing interface as we will see in the following section.



Why is each phase important?

When writing a custom report you often have a heading section at the top, a summary section at the end, and probably a data-centric section in the middle. We are going to organize our ODS report writing code into each of these respective sections so that they are conditionally executed.

OBJECT-ORIENTED CONCEPTS

If you have never used an object-oriented programming language before, you will need to learn a few basic concepts before you can begin writing code.

- What is an object?

It is a self-sufficient module whose purpose is to perform a set of related tasks. By encapsulating common tasks around a common data structure, the architecture promotes code reusability. An object contains both data structures, and application programming interface (API) that know how to perform actions utilizing the common data structures.

- What is a method?

It is another name for the objects application programming interface (function definition). A method can define its own required/optional arguments to perform an action (you can think of a method as a verb). A method can be called *only* by its own object. This is one of the main distinctions between a traditional functional paradigm and an object-oriented method paradigm.

- What is a class?

A class is a formal declaration of the characteristics (attributes or data structures) and actions (methods) that an object may perform. For example, a class named Cars would consist of characteristics shared by all cars such as make, model, year, color, and manufacturer, and may have the ability to perform actions such as drive, break, honk, or turn (methods). So in our hypothetical class, the action "drive" would be considered a method, and I may want to provide the "drive" method with additional instructions on how to drive. These instructions would be our method arguments such as forward, backward, or speed.

- What does it mean to create an instance of a class?

This is the runtime initialization of the class object attributes and methods. Now you can call methods!!!

APPLYING OBJECT-ORIENTED CONCEPTS TO THE DATA STEP

The DATA step has created a new statement that will allow you to create an object of a given class.

```
declare < class name > < local variable name >;
```

Here is how you would declare an ODS Report Writing class (odsout) object and assign it to a local variable named "ODS".

```
declare odsout ODS;
```

We have not created a runtime instance (or initialized) of the local variable "ODS" yet. This can be done in two ways. This is really more a personal preference.

```
ODS = _new_ (odsout);
```

Or by combining the declaration:

```
declare odsout ODS();
```

In the previous section, we established that the body of the DATA step is executed once for each input observation during the execution phase. Unlike DATA step variables that can be re-initialized for each input observation, the local object variable "ODS" needs to be initialized only once for the life of the DATA step. Let's instantiate the variable "ODS" on the first input observation as follows.

```
if _n_ = 1 then do;
    declare odsout ODS();
end;
```

USING THE EXECUTION PHASES TO ORGANIZE YOUR OUTPUT

- How can I tell if I am in the initialization execution phase?

When we are executing the observation loop for the first time, then we are in the initialization execution phase. When publishing custom reports, there may be specific method calls that we want to execute only at the beginning. For example, to add a special heading, start a table, or write an introductory paragraph such as...

```
if _n_ = 1 then do;
    declare odsout ODS();
    ODS.format_text(data: "You are in the initialization execution phase!!!");
end;
```

- How can I tell if I am in the termination execution phase?

When we execute the observation loop for the very last time, then we are in the termination execution phase. Very often, this phase is used to add a special footer, end a table that may have been started during the initialization phase, or summarize a closing argument. To detect the termination execution, phase use the END= data step option.

```
data _null_;
set sashelp.orsales (end=eof);

if eof = 1 then do;
    ODS.format_text(data: "You are in the execution phase!!!");
    ODS.delete();
end;

run;
```

USING DATA STEP BY-GROUP PROCESSING TO ORGANIZE YOUR OUTPUT

One of the purposes of using the ODS Report writing interface is to create custom reports that are dynamically generated from real data such as an invoice. Consider that I have a database of customers that contain invoice information. I want to create an automated DATA _NULL_ program that uses my customer database as an input data set to create one invoice (PDF file) for each customer containing all of the appropriate billing information for that specific customer. By taking advantage of the built in BY-group processing the DATA step provides, this task can easily be accomplished, and the best part is that the data is dynamically driving the generation of the report. Next month when the number of customers has changed, and the account information has changed, the program

continues to produce invoices indefinitely until you want to modify the structure of the report. As a matter of fact, we can use WHERE processing to subset the data to include only those customers that may be 30, 60, or 90 days past due. We will examine a complete invoice example after we complete our introduction of the ODS report writing methods. For a detailed explanation of the DATA step BY-group processing, please see the “BY-Group Processing in SAS Programs” chapter in the *SAS Language Reference: Concepts*.

- How does BY-group processing aid the ODS Report Writing Interface?

It allows the program to take advantage of additional data-driven built-in logic. The DATA step identifies the beginning and ending of each BY group by creating additional temporary variables: FIRST.<BY-group variable> and LAST.< BY-group variable>. During our observation loop processing if we are on the first observation of a BY group, the value of FIRST.<by group variable> will be set to 1. Likewise during our observation loop processing if we are on the last observation of a BY group, LAST.<BY-group variable> will be set to 1. Here is a simple example using some pseudo code to illustrate what BY group processing would look like.

```
data _null_;
set customers;
by account_number;

if first.account_number then do;
  < start a new invoice (PDF) file >
  < write out heading information >
end;

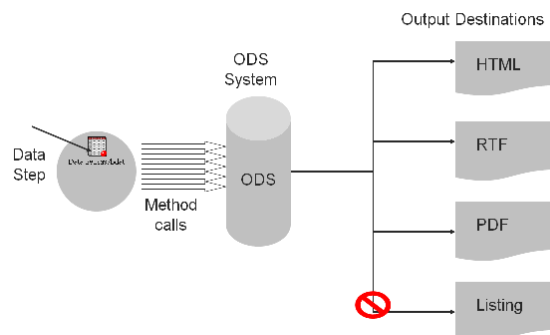
< all the ODS Report writing code to generate the body of the invoice >

if last.account_number do;
  < write out footer information >
end;
run;
```

METHOD CATEGORIES

There are various methods that comprise the ODS Report Writing Interface, but they can be broken down into five unique categories of features such as producing text, tables, controlling the page, arranging output (Layout), as well as some more general-purpose methods. We will explore the syntax of each unique category separately first by reviewing the syntax, and then applying the features in an example. Before we get to looking at the specific syntax of a particular method call, let's explore the process of a method call.

Method Process Flow



Up until now, the PROC has been the one driving the ODS system to produce output in each active output destination. The ODS Report Writing interface simply gives the programmer complete access to the ODS System and allows the DATA step to communicate directly with the fundamental pieces of the ODS Architecture. The programmer can use the DATA steps conditional logic to control how to instruct ODS to present the information.

NOT ALL OUTPUT DESTINATIONS ARE CREATED EQUAL

There are various features that the ODS Report Writing interface relies on in order to fully support all of the documented methods; however, not all output destinations support all features. The report writing interface is intended for those ODS destinations that produce high-quality two-dimensional reports. Traditional data _NULL_ report writing interface provides a low-quality fixed column/row (cell) wise reporting interface that is specifically designed to work with the ODS listing destination. You will find that not all destinations support the same set of features, and therefore some methods may have restricted support in some destinations.

TEXT METHODS

Methods in the text category are intended to provide you with the ability to produce high-quality formatted text. Whether publishing a form letter, invoice, or just adding narrative text to the body of your document, you will find an elaborate list of customization features to convey the importance of your message.

- Format_text Displays active text
- Note Displays text as a note

USING TEXT METHODS

The text methods can provide you with a rich set of text-formatting features whether your text is static or being provided from a data source. In the following example, we are going to format text coming from a SAS data set called Brand that has two variables “tagline” and “description,” and we are going to display this data set in a non-rectangular manner using our new text formatting methods.



Figure 1.1. Using Text Methods

Let's look at the DATA step-specific portion of the code. The entire program is contained in Appendix 1, Example 1.1.

```
data _null_;  
set brand end=eof;
```

Notice that while processing the first observation (`_n_ = 1`), I create an instance of the “odsout” class and assign it to the local variable “obj”. I also create a note using some static text, and apply some style attributes to customize the font and color. I add an image to my text as well.

```

if _n_ =1 then do;
  declare odsout obj();
  obj.note(data: "Our Brand...",
    overrides: "preimage='c:\Public\images\star.gif'
              font_style=italic
              font_size=12pt
              font_weight=bold
              color=cxbbb2e0",
    just: "L");
end;

```

These FORMAT_TEXT methods are not surrounded by any conditional logic, so they will be executed for each observation in the input data set. We are simply displaying the value of the “tagline” and “description” variables with some custom formatting characteristics. Notice that each FORMAT_TEXT method will produce an entire line in the output. You are free to use DATA step string concatenation, or provide additional formatting through the use of a SAS format.

```

obj.format_text(data: tagline,
  overrides: "background=cx494068
            color=cxbbb2e0
            font_size=12pt
            font_style=italic
            width=2.5in",
  just: "C");
obj.format_text(data: description,
  overrides: "background=cxbbb2e0
            font_style=italic
            font_size=8pt
            width=2.5in",
  just: "C");

```

I want to insert a blank line in between each tagline and description as long as it is not the last observation because that would add a blank line to the end of my report. So here we are using the END= DATA step option to display the blank line only when desired.

```

if eof ne 1 then
  obj.format_text(data: " ",
    overrides: "height=1mm");
;RUN;

```

TABLE CATEGORY

Methods in this category are intended to create tabular output that consists of row/column headers, and data values in a structured format. PROCs automatically produce tabular output to display the results of the analysis requested, however you only have limited control of the content displayed. Please see “The TEMPLATE Procedure” section in the ODS documentation to learn more about modifying your procedure output. The ODS Report Writing interface provides the ability to create tabular output a single cell at a time while providing an endless number of formatting capabilities. This interface is specifically intended to address the formatting limitations of the TEMPLATE procedure as well as the limitations of the standard report-writing procedures. Before we get to looking at the specific syntax of the table methods, let’s examine the anatomy of a table.

- What is a table?

It is a 2-D arrangement of information that is traditionally displayed in a manner in which similar categories or classes of information are displayed in a structured manner (a collection of rows).

- What is a row?

The horizontal dimension of a 2-D arrangement (a collection of cells in the horizontal dimension).

- What is a column?
The vertical dimension of a 2-D arrangement (a collection of cells in the vertical dimension).
- What is a cell?
A collection of data, text, or images that can span multiple rows or columns.
- What is cell padding?
The thickness on each of the four sides of the cell.
- What is cell spacing?
The thickness of the spacing between cells.

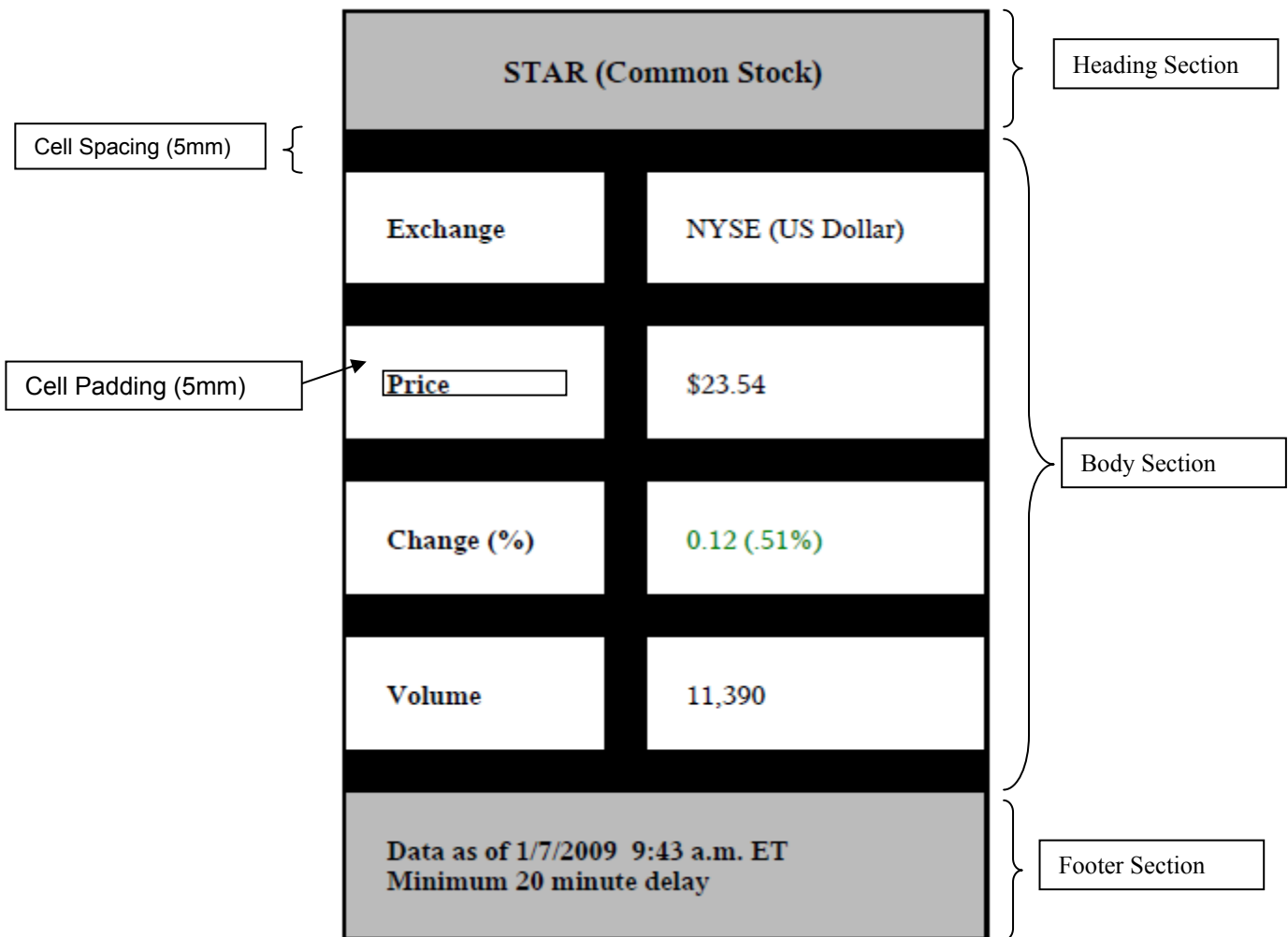


Figure 1.1. The Anatomy of a Table

TABLE SECTION METHODS

- HEAD_START Start the Table Heading Section.
- HEAD_END End the Table Heading Section.
- BODY_START Start the Table Body (Data) Section.
- BODY_END End the Table Body (Data) Section.
- FOOT_START Start the Table Footer Section.
- FOOT_END End the Table Footer Section.

TABLE METHODS

Data is not always presented in a strict Excel-like fashion (1 observation => 1 row). Often, you may want to show one observation in multiple rows/columns format, or multiple observations in a single row/column format. Once you start a table, you will want to continue using the table methods exclusively until you close the table. The only exception is that `FORMAT_TEXT` can be used in conjunction with the `CELL_START` and `CELL_END` methods as we will see in our following example.

- TABLE_START Start a table.
- TABLE_END End a table.
- ROW_START Start a row.
- ROW_END End a row.
- CELL_START Start a cell.
- CELL_END End a cell
- FORMAT_CELL A more convenient way to create a cell.

USING TABLE METHODS

The table methods allow you to build a table one cell at a time. This approach is invaluable when a custom table format is desired. In this example, I have hard-coded all of the data values just for convenience, but the data could have been supplied via an input data set. Notice that there is no conditional logic at all and no input data set; therefore, the code is executed just one time.

STAR (Common Stock)	
Exchange	NYSE (US Dollar)
Price	\$23.54
Change (%)	0.12 (.51%)
Volume	11,390
Data as of 1/7/2009 9:43 a.m. ET Minimum 20 minute delay	

Provided to you compliments of SAS 9.2 using ODS Report Writing Interface using Table methods.

Figure 1.2. Using Table Methods

Let's look at the DATA step-specific portion of the code. The entire program is contained in the Appendix 1, Example 1.2.

```
data _null_;
dcl odsout obj();
```

Start a table by giving it a name="Stock" and a label="Stock Quote" which will be the corresponding leaf node name and label that will be used in the DMS Results window as well as in the table of contents. We also specify that the overall table maximum width is 4 inches. If the width was not specified, the table would measure each cell and determine the maximum width for you as long as it does not exceed the physical page restrictions.

```
obj.table_start(name: "Stock",
               label: "Stock Quote",
               overrides: "width=4in");
```

Remember, "tables" can contain only "rows," and "rows" can contain only "cells." Once the first row is closed, the maximum number of cells has been determined. In this next section, I start a row and create a single cell. However, notice that I told the cell to span two columns. So, in effect it really is two cells. However, I have allowed the data to be displayed as if it were a single cell.

```
obj.row_start();
obj.format_cell(data: "STAR (Common Stock)",
               column_span: 2,
               overrides: "backgroundcolor=cx494068 color=white");
obj.row_end();
```

In rows 2–4, I create two separate cells individually using the FORMAT_CELL method providing various style attribute overrides to control color, font characteristics, and justification. In a more elaborate example, it would be common to use the DATA steps conditional logic to apply different characteristics based on your data values. For example, I may want to conditionally color the change value based on whether it is negative (red), or positive (green).

```

obj.row_start();
  obj.format_cell(data: "Exchange",
                 overrides: "just=left font_weight=bold");
  obj.format_cell(data: "NYSE (US Dollar)",
                 overrides: "just=left");
obj.row_end();

obj.row_start();
  obj.format_cell(data: "Price",
                 overrides: "just=left font_weight=bold");
  obj.format_cell(data: "$23.54",
                 overrides: "just=left");
obj.row_end();

obj.row_start();
  obj.format_cell(data: "Change (%)",
                 overrides: "just=left font_weight=bold");
  obj.format_cell(data: "0.12 (.51%)",
                 overrides: "just=left color=green");
obj.row_end();

```

In row 5, instead of using the FORMAT_CELL method I use the equivalent CELL_START, FORMAT_TEXT, and CELL_END approach. This is normally unnecessary, but there may be special cases where this approach can be beneficial.

```

obj.row_start();
  obj.cell_start();
  obj.format_text(data: "Volume",
                 overrides: "just=left font_weight=bold");
  obj.cell_end();
  obj.format_cell(data: "11,390",
                 overrides: "just=left");
obj.row_end();

```

In row 6, I once again instruct the FORMAT_CELL method to span multiple columns, but notice that the text is drawn on two separate lines. The split character argument instructs ODS to perform a new line whenever the provided character is encountered. Notice that the "*" character is not displayed. This is treated as an instruction instead of a literal character. Finally, we close the table by using the TABLE_END method.

```

obj.row_start();
  obj.format_cell(data: "Data as of 1/7/2009 9:43 am ET *Minimum 20 minute delay",
                 column_span: 2,
                 split: '*',
                 overrides: "font_size=10pt just=left");
obj.row_end();
obj.table_end();
;run;

```

PAGE METHODS

Page-related methods control characteristics of the entire page. For example, you can arbitrarily start a new page or modify an existing title or footnote. The TITLE and FOOTNOTE methods will allow you to incorporate a particular data value into your title or footnote. These methods are intended to work similarly to the global TITLE and FOOTNOTE statements available outside the DATA step.

- PAGE Start a new page.
- TITLE Add, delete, or update a title in the system.
- FOOTNOTE Add, delete, or update a footnote in the system.

LAYOUT CATEGORY

When the placement of output on a page necessitates a detailed level of control, you will find that the layout methods may be a more suitable solution. Despite being preproduction in SAS 9.2, you will find many enhanced features that are designed to provide you with unlimited control of your page. Layout has two distinct forms: Absolute and Gridded; both have unique strengths and weaknesses. The most important decision when using ODS Layout is in selecting the type that best suits your report design. ODS Layout is designed to allow nested layouts (containers) to provide endless customization. You are not limited to a single Layout type; a gridded layout can contain absolute layouts, and vice versa. This is where adequate design work will greatly benefit your coding efforts.

NOT ALL OUTPUT DESTINATION ARE CREATED EQUAL

There are a variety of limitations imposed on ODS Layout by the destination itself. For example, the PDF (or PRINTER) destinations are all limited to the physical page size of the piece of paper that we are writing to. In the case of HTML, there really is no concept of a physical page size or an X or Y position because the browser is managing the placement of the output in relation to the size and placement of the browser window. Due to some of these external restrictions that are being imposed on our report design, we may get varying results when using differing output destinations. However, you will find that we will always make an effort to provide the most intuitive defaults when dealing with these limitations and will make every effort to produce reasonably similar-looking output.

ODS LAYOUT TERMINOLOGY, CONCEPTS, AND THINGS TO CONSIDER

Layout Container Is an area that contains a collection of regions. Layouts can contain only regions. Layout can have a fixed size (like width=3in and height=4in) or can be dynamically sized to accommodate the child regions.

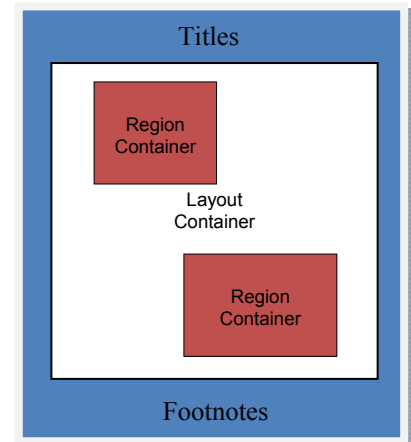
Region Container Is an area that contains output (like text, tables, graphs), or nested layout containers. Regions can also have a fixed size, or may be dynamically sized to accommodate the collection of child output.

Dimension Unit Is a nonnegative number, optionally followed by one of the following units of measure. A number without a unit of measure will be interpreted as pixels. It is not recommended that you size things in pixels because of adverse dependencies on resolution that can differ between destinations.

cm	centimeters
em	standard typesetting measurements unit for width
ex	standard typesetting measurements unit for height
in	inches
mm	millimeters
pt	a printer's point

Titles & Footnotes Global titles and footnote are placed outside of the outermost layout container. Title and footnote processing is always done before any output is produced on the physical page. This is often referred to as page initialization time.

PROCTITLES PROCTITLES are placed inside the region adjacent to the PROC output.



GRIDDED LAYOUT

Gridded layout allows you to arrange output in a two-dimensional relative grid structure (like a spreadsheet or piece of graph paper), and is the preferred approach to managing output on the page. Gridded layout is a much simpler mechanism for arranging output. Yet at the same time, it also is a much more powerful alternative than absolute layout. Gridded layout address all of the limitations that absolute layout has to offer such as enforcing automatic alignment of respective grid cells, the ability to have a layout continue onto the next page if necessary, the ability to dynamically compute the size of a grid cell, and the ease in which you can maintain the integrity of the report(or program).

Valid Destinations: HTML and PRINTER destinations.

GRIDDED LAYOUT METHODS

- LAYOUT_GRIDDED Create a gridded layout container.
- REGION Create a region container.
- LAYOUT_END End a gridded layout container

USING GRIDDED LAYOUT METHODS

When a more detailed level of placement is necessary, the layout methods will be your best solution. In the following example, we will experiment with arranging images and text in a grid-like fashion. Our data is provided from a SAS data set that has three variables: PHOTO (employee photo), NAME (employee name), and BIO (employee biography). Almost any organization has a company employee database. An employee database has unique data characteristics that are less traditional than reporting data sources like inventory levels or sales data. Because of the unique nature of an employee database, traditional reporting approaches are probably not going to be well-suited for presenting this information. The ODS Report Writing interface provides you with the flexibility to address these custom report-writing requirements. Let's take a look at how we generated the following report.

The screenshot shows a report header for 'ORION STAR Sports & Outdoors' with the title 'Our Founders'. Below the header, there are two rows of content. The first row features a portrait of Scott Huntley, followed by a text block stating he is the CEO and President of Orion Star, co-founded in 1976. The second row features a portrait of Dan OConnor, followed by a text block stating he is the CIO of Orion Star, which he also helped co-found in 1976. At the bottom of the report, there is a footer note: 'Provided to you compliments of SAS 9.2 using ODS Report Writing Interface Gridded Layout features.'

Figure 1.3. Using Gridded Layout Methods

Let's look at the DATA step-specific portion of the code. The entire program is contained in the Appendix 1, Example 1.3.

```
data _null_;  
set execs end=eof;
```

Notice that while processing the first observation (`_n_ = 1`) I create an instance of the "odsout" class and assign it to the local variable OBJ, and create a gridded layout that has two columns. I also have instructed the gridded layout to center itself horizontally on the page.

```

if _n_=1 then do;
  dcl odsout obj();
  obj.layout_gridded(columns: 2,
                    overrides: "just=center");
end;

```

Remember, a “layout” can contain only “regions”, and only “regions” can contain output (tables, text, images, or graphs). I create my first region which will correspond to column 1 in my gridded output, and place the current observations of employee photo and name in our first region (column). Notice that I provide a width for the first region which will provide a maximum width for all output that will be rendered. If I had not provided a width for the region it would have measured the employee photo and name and would dynamically size the region to be the maximum width of the two items.

```

obj.region(width: "1.25in");
obj.image(file: photo,
         overrides: "just=left width=100pct");
obj.format_text(data: strip(name),
               overrides: "just=left width=100pct");

```

Remember: When in a gridded layout, regions are populated similar to the way a table is produced from left → right and top → bottom. Therefore, when I create the second region, it is arranged to the right of the first region because my gridded layout is made up of a two-column grid. Notice that I also provide a width for the second region which will provide a maximum width for all of the output rendered. We have established that our gridded layout is structured in two-column grid with the first region having a width of 1.25 inches and a second region having a width of 3.25 inches which means that our entire layout size can now be calculated to be 4.5 inches. In this example, I have set the paper size to be 4 inches tall by 6 inches wide. Therefore to center the layout as specified in our “layout_gridded” method, we will have 0.75 inches to the left of the table as well as 0.75 inches to the right. The really cool thing is that it is all done for you!!

```

obj.region(width: "3.25in");
obj.format_text(data: strip(bio));

```

After displaying the bio in region 2, I once again take advantage of the END= DATA step option to allow me to insert a blank line between each employee’s information unless I am processing the last employee in which case I need to close my gridded layout. You can start to see how the arrangement of your method call really dictates the structure of your output, and how using the DATA step’s conditional logic provides you with an almost endless list of possibilities.

```

if eof ne 1 then
  obj.format_text(data: "");
else
  obj.layout_end();
;run;

```

ABSOLUTE LAYOUT

Absolute layout allows you to specify the exact location on the page to place a layout and region container. This approach is very precise, but often somewhat error-prone. The sole responsibility for correctly placing each container falls on the programmer. Often, this creates an overly complicated program because each container needs to be explicitly placed to ensure no overlap, and programmatic alignment is retained. This creates a code management nightmare because, if one container’s position is altered, you may need to manually alter all containers to maintain your report’s integrity. Absolute layout is also restricted to a single page. If the output is too large to fit in the fixed size container, the output will be discarded, and you will receive a blank region and a warning in your log. With all that doom and gloom out of the way, absolute layout does have its merits. Absolute layout is perfectly suitable for very

static types of output, placing output in a specific location on a pre-printed form, creating cover pages, or precisely placing output in a nested region container.

Valid Destinations: PRINTER destinations only (like PDF, PS, PCL).

ABSOLUTE LAYOUT METHODS

- LAYOUT_ABSOLUTE Create an absolute layout container.
- REGION Create a region container.
- LAYOUT_END End an absolute layout container.

USING ABSOLUTE LAYOUT METHODS TO CREATE A PROMOTIONAL MAILER

Whether it is a promotional mailer, customer targeted letter, or part of a promotional email campaign, the combination of static content and data-specific content is pervasive in any business today. The ODS Report Writing interface is perfectly suited for producing this type of content. Consider that Orion Star wants to execute a customer loyalty program for all of its current Club Members by sending a “25% off” flyer. Each flyer needs to be individually addressed to each customer, and could be tailored specifically to each customer’s preferred shopping preference by using the DATA step conditional logic. However for the purposes of this example I am going to create a generic flyer that creates a separate PDF file for each customer who could be either snail-mailed or e-mailed. Our data is provided from a SAS data set of Orion Star’s Club Members and contains “name”, “address”, “email”, and “preferredcategory” variables.

ORION STAR
Sports & Outdoors

Private Sale
Now through March 25
25% off
your entire purchase when you bring this card

Excludes previously discounted items, layaways and prior sales. Cannot be combined with any other offer.

No payments for 12 months*

*on purchases over \$500 with your Orion Star preferred Account. The Orion Star Preferred Account is subject to credit approval as determined by the lender Outdoor Bank, Burlington, Vermont, is available to qualified US residents, and is governed by Vermont and Federal Law. The promotion End Date for this offer is the last day of the calendar month which is twelve months from the date of purchase. If balance is not paid in full by the Promotion End Date show on your billing statement, Finance Charges will accrue from the transaction date at an ANNUAL PERCENTAGE RATE of 21.99%. A Minimum Finance Charge of \$2.00 will apply. Offer valid in Orion Star stores only and expires 3/31/2009.

Pleasant Valley Promenade
6204-121 Glenwood ave.
Raleigh, NC
919.432.0987

The streets at Southpoint
3454 Executor Way
Durham, NC
919.320.6238

Crabtree Valley Mall
Raleigh, NC
919.432.0987

Store Hours
Monday - Saturday
10am - 8pm
Sunday
Noon - 5pm

Free Gift Wrapping

Exclusive Invitation for our Club Members

David Kelley
507 Down Patrick Lane
Gamer, NC 27644

Figure 1.4. Using Absolute Layout Methods

Let’s look at the DATA step-specific portion of the code. The entire program is contained in the Appendix 1, Example 1.4. Absolute layout methods are probably the least stable of all of the ODS Report Writing interface.

To create a separate PDF file for each customer in our input database, we need to use a couple of features, the first being the newfile= option on the ODS PDF statement. This tells ODS to create a new pdf file for each By-group (or each customer). To do BY-group processing in the DATA step, you use the “bv name:” statement.


```
ods pdf file="Promotion.pdf" notoc newfile=Bygroup;
```

```
data _null_;  
set customers end=eof;  
by name;  
if _n_ =1 then do;  
  declare odsout obj();  
end;
```

There is one final step in creating a new PDF file for each customer in our input data set, and that is to notify ODS that we are starting a new By-group which can be done by using the OPEN_DIR method with the special BY: 1 argument. I am also creating a directory in my DMS Results window using the NAME and EMAIL data variables.

```
obj.open_dir(name: name,  
            label: email,  
            by: 1);
```

Creating an absolute layout it easy. Notice that I have not specified a fixed size for my absolute layout so the dimension will be dynamically calculated based on the encapsulated regions; however, I do specify that I want a pixel border around the layout.

```
obj.layout_absolute(overrides: "borderwidth=1");
```

My first absolute region has specified a HEIGHT and WIDTH with additional instructions to change the background color to be "cx494068". We then proceed to insert an image, some static text, and a nested gridded layout.

```
obj.region(width: "2.4in",  
          height: "4in",  
          overrides: "background=cx494068");  
obj.image(file: "c:\Public\Images\orionstarHeader.jpg");  
obj.format_text(data: "Private Sale",  
               overrides: "color=cxbbb2e0 font_size=32pt",  
               just: 'c');  
obj.layout_gridded(columns: 3,  
                  row_gutter: "1mm",  
                  column_gutter: "1mm");  
obj.region();  
  obj.format_text(data: "25",  
                 overrides: "color=cxbbb2e0 font_size=72pt");  
obj.region();  
  obj.format_text(data: "%*off",  
                 split: "*",  
                 overrides: "color=cxbbb2e0 font_size=32pt");  
obj.region();  
  obj.format_text(data: "Now *through *March 25",  
                 split: "*",  
                 overrides: "color=cxbbb2e0 font_size=10pt");  
obj.layout_end();  
obj.format_text(data: "your entire purchase *when you bring this card",  
               split: "*",  
               overrides: "color=cxbbb2e0 font_size=12pt width=85pct",  
               just: 'C');  
< more... >
```

I Create a second region which contains multiple tables of information. This region contains local store information, store hours, and a special free wrapping service offered to club members only. If this were a real-world example, you almost certainly would use the DATA step's conditional logic to include only those stores that are close to our customer's home address. Unlike our gridded regions that are positioned relative to each other, the programmer is required to specifically place each region at a given location on the page so as to not obscure any other output.

```

obj.region(x: "2.4in",
          width: "1.25in",
          height: "4in",
          overrides: "background=cx494068 borderwidth=1");
obj.table_start(overrides: "rules=none frame=void cellpadding=0 cellspacing=0");
obj.row_start();
obj.format_cell(data: "Pleasant Valley Promenade",
               overrides: "color=cxbbb2e0 font_size=6pt font_weight=bold");
obj.row_end();
obj.row_start();
obj.format_cell(data: "6204-121 Glennwood ave.",
               overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Raleigh, NC",
               overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "919.432.0987",
               overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "", overrides:"font_size=6pt");
obj.row_end();
< more... >

obj.table_start(overrides: "frame=box");
obj.row_start();
obj.format_cell(data: "Free Gift Wrapping",
               overrides: "color=cxbbb2e0 font_size=10pt just=center
                           vjust=middle width=1in height=.5in font_weight=bold");
obj.row_end();
obj.table_end();

```

The third and final region of this example presents the data specific portion of our promotional flyer. Notice here instead of passing quoted strings of static text I am passing the actual input data variable values. The DATA step also has built in string concatenation logic which also allows us to concatenate the city, state, and zip.

```

obj.region(x: "1.9in",
          width: "2.75in",
          height: "4in");
obj.format_text(data: "Exclusive *Invitation *for our *Club *Members",
               split: "*",
               overrides: "color=cx494068 font_size=36pt");
obj.format_text(data: " ");

obj.format_text(data: name);
obj.format_text(data: street);
obj.format_text(data: strip(city) || ", " || state || " " || zip);

```

We are creating an entire page for each observation processed. I will want to close the absolute layout and directory that was created for this observation so that the next loop through we can start our new flyer for the next club member.

```

obj.layout_end();
obj.close_dir();
;run;
ods pdf close;

```


I used a variety of ODS Report Writing methods in each of these regions to show the flexibility of this programming interface. The absolute layout methods only provided region containers to organize the output that we wanted to display irrespective of whether it was a table, image, text, or another nested layout.

MISCELLANEOUS CATEGORY

- OPEN_DIR Insert a custom folder in the table of contents and DMS Results window.
- CLOSE_DIR Close a custom folder in the table of contents and DMS Results window.
- LINE Draw a horizontal line across the page.
- IMAGE Insert an image.
- HREF Create a URL link that will be taken when the item is clicked.
- DELETE Delete the object.

CREATING A FORM LETTER

The content of a form letter can vary greatly from completely static text to largely data-dependent information. In the following example, I have a good mix of both. Once again, I am using my fictitious Orion Star Sports & Outdoor retailer to create a form letter for a customer loyalty program called the “Orion Star Club” rewards program. The customer will receive one point for every dollar spent, and an additional one point for every dollar spent using his or her “Orion Star Club Visa” card.

This example uses three different data sources that will contribute to the form letter using a single DATA_NULL_ program.

1. customer data set that contains name, address, and account number information
2. product data set that contains product descriptions, photos, pricing, and product ID information
3. transaction data set that contains invoice and purchasing information

ORION STAR
Sports & Outdoors

Preferred Club Member

SAS Campus Drive
Cary, NC 27513
February 11, 2009
Dear Dan O'Connor,

As a preferred Club Member you receive special membership benefits such as special promotions, member discounts, free 1 year warranty on all purchases, as bonus points that can be redeemed for free items.

See what being a preferred Club Member is all about.
You are pre-approved for our new Orion Star Club Visa card.

1. A low 8.9% APR
2. Get double membership points when using your Orion Star Club Visa card.
3. Get Exclusive access to offers not available to general public.

[Apply Today](#)

Current Club Point Balance		256
Recent Purchases		
Date	Invoice Number	Additional Club Points
01/02/09	1	360
Bonus Visa Club Points		360
01/05/09	2	200
New Club Point Balance		1,176

How to Redeem Points:
Step-by-step instruction on how to use your Club REWARD Point to purchase free items.

[Learn How](#)

Provided to you compliments of SAS 9.2 using ODS Report Writing Delivery methods.

Figure 1.5. Creating a Form Letter

Let's look at the DATA step-specific portion of the code. The entire program is contained in Appendix 1, Example 1.5. In this example, I am going to use another object-oriented class called the "data step hash object." For more information on how to use the hash object, please see <http://support.sas.com>.

Use the DATA Step retain statement to retain summarization information across multiple purchase transactions (observations) to show the monthly accumulated club points for each customer.

```
data _null_;
retain invsum bonus sum clubpoints;
```

On the first input observation, create two hash objects, one with a key index value using the customer's "accountid" to look up customer information and a second one with a key index using the "productid" to look up pricing inventory information for each transaction. This step simply creates the lookup hash object.

```
if _n_ =1 then do;
declare odsout obj();

length name $20 street $22 city $12 state $3;
zip=.;
declare hash customer(dataset: "customers");
customer.defineKey("accountid");
customer.defineData("name", "street", "city", "state", "zip", "clubpoints");
customer.defineDone();
```

```

length item $64;
itemno=.; instock=.; productstatus=.; regprice=.; saleprice=.; additionalshipping=.;
declare hash product(dataset: "inventory");
product.defineKey("itemno");
product.defineData("item", "instock", "productstatus", "regprice", "saleprice",
"additionalshipping");
product.defineDone();
end;

set transactions end=eof;
by accountid invoice;
if first.accountid then do;
sum=0;
obj.open_dir(name: "Invoice",
label: "Invoice for Customer " || accountid,
by: 1);

```

Seeing that my input transaction data set shares common variables “accountid” and “productid” with the keys in my lookup hash objects, if I call the find method, it automatically uses the current “account id” value as a lookup. If the “accounted” value is found in the hash object any variables defined using the “defineData” method will be accessible. Use the gridded layout and FORMAT_TEXT methods to display the customer address information, static text information describing the program, and offering a special Visa Application.

```

if customer.find() eq 0 then do;
obj.layout_gridded(columns: 2);
obj.region(width: "3.25in");
obj.format_text(data: strip(street),
overrides: "font_size=14pt width=100pct just=left");
obj.format_text(data: strip(city) || ", " || state || " " || zip,
overrides: "font_size= 14pt width=100pct just=left" );
obj.format_text(data: " ");
obj.format_text(data: put(today(), worddate18.),
overrides: "font_size=14pt width=100pct just=left");
obj.format_text(data: " ");
obj.format_text(data: "Dear " || strip(name) || ",",
overrides: "font_size=14pt width=100pct just=left");
obj.region();
obj.image(file: "c:\public\Images\Scenic2.jpg");
obj.layout_end();

< more . . .>

```

Here, we are creating a table that shows the customer’s previously accumulated club points, as well as any points that have been accumulated during this month’s billing cycle with purchase date, invoice number, credited points for each invoice. I also conditionally add a “Bonus Visa Club Point” row to the table if the customer has paid for these items with their Orion Star Cub Visa card. Notice that this is completely conditional based on my input data.

```

obj.table_start(overrides: "width=100pct");
obj.row_start();
obj.format_cell(data: "Current Club Point Balance",
column_span: 2,
overrides: "font_weight=bold font_size=16pt");
obj.format_cell(data: clubpoints,
format: "comma8",
overrides: "font_weight=bold font_size=16pt");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Recent Purchases",
column_span: 3 );
obj.row_end();

```

```

obj.row_start();
obj.format_cell(data: "Date");
obj.format_cell(data: "Invoice Number");
obj.format_cell(data: "Additional Club Points");
obj.row_end();
end;
end;

```

On the first BY observation of an invoice, I want to clear my retained variable so that I can sum the club points for each invoice.

```

if first.invoice then do;
  invsum=0;
  bonus=0;
end;

```

My transaction dataset only contains the “productid” of the items that was purchased so I have to use my product hash object to look up the product description, and price information so that I can correctly calculate the correct number of club points to credit to the customer. Isn’t this cool?!

```

if product.find() eq 0 then do;
  if saleprice ne 0 then
    cost = quantity*saleprice;
  else
    cost = quantity*regprice;
  invsum = invsum + cost;
  sum = sum + cost;
end;

```

If the customer used their Orion Star Club Visa card, calculate the bonus club points.

```

if clubvisa eq 1 then do;
  bonus = bonus + clubvisa*cost;
  sum = sum + clubvisa*cost;
end;

```

When we have reached the last observation for this particular form letter display rows showing the summarized club and bonus points.

```

if last.invoice then do;
  obj.row_start();
  obj.format_cell(data: date,
    format: "mmdyy10.");
  obj.format_cell(data: invoice);
  obj.format_cell(data: invsum,
    format: "comma8.");
  obj.row_end();

  if clubvisa eq 1 then do;
    obj.row_start();
    obj.format_cell(data: "Bonus Visa Club Points",
      column_span: 2);
    obj.format_cell(data: bonus,
      format: "comma8.");
    obj.row_end();
  end;
end;

```

When we exhausted all the invoices for the current customer, display the “New Club Point Balance”, and provide instruction on how to redeem the points.

```

if last.accountid then do;
  obj.row_start();
  obj.format_cell(data: "New Club Point Balance",
    column_span: 2,
    overrides: "font_weight=bold font_size=16pt");
  obj.format_cell(data: clubpoints+sum,
    overrides: "font_weight=bold font_size=16pt",
    format: "comma8.");
  obj.row_end();

obj.table_end();

obj.format_text(data: " ",
  overrides: "font_size=16pt");
obj.format_text(data: "How to Redeem Points?",
  overrides: "background=cxbbb2e0 font_size=20pt");
obj.format_text(data: "Step-by-step instruction on how to use your Club REWARD Point
to purchase free items.",
  overrides: "font_size=16pt");
obj.format_text(data: " ");
obj.format_text(data: "~{style [font_size=20pt
url='http://www.orionstar\redeempoints']Learn How!}");



  obj.close_dir();
end;

if end eq 1 then do;
  obj.delete();
  customer.delete();
  product.delete();
end;
;run;

```

CREATING A CUSTOMER INVOICE

Generating invoices can be similar to creating form letters; however, invoices traditionally are much more data-centric, and may take advantage of more elaborate conditional logic. Using the DATA step, you can create a more tailored invoice for each customer. This could also create an opportunity to do some suggestive selling based on the customer's recent purchasing behavior. This is where the DATA step functionality will allow you to be more creative than those traditional canned template approaches and hopefully yield a better ROI on your promotional activities.

ORION STAR Sports & Outdoors		Invoice		
Dan O'Connor		Invoice Number: 1		
SAS Campus Drive				
Cary, NC 27513				
January 2, 2009				
Items(s)	Quantity	Regular Price	Sales Price	Total
 Orion Star King Hang-On Tree stand	1	89.99	79.99	\$79.99
 Orion Star XT Mega-Sized Binoculars	1	279.99	0	\$279.99
			Subtotal	\$359.98
			US Shipping	\$12.00
			Tax	\$28.00
			Total	\$400.78

Provided to you courtesy of SAS R.2 using ODS Report Writing Interface methods.

Figure 1.6. Creating an Invoice

Let's look at the DATA step-specific portion of the code. The entire program is contained in Appendix 1, Example 1.6. In this example, I am also going to use the DATA step hash object as I did in Example 1.5. If you would like to see detailed information on how to use the hash object, please see <http://support.sas.com>. Seeing that I detailed the use of the hash object in Example 1.5, I am also going to eliminate that code from this discussion.

Use the DATA Step retain statement to retain summarization information across multiple purchase transactions (observations) so that we can display all the items purchased, price, shipping, and total for each invoice.

```

Data _null_;
retain foundcustomer sum shipping;

< more ... >

```

We are using the same input data set as in example 1.5 that contains product sales information for this month's billing cycle for each invoice.

```

set transactions end=eof;
by invoice;

```

On the first observation of a new invoice transaction, I want to clear my retained variables so that I can accurately compute product sales information for each product purchased on this particular

```

if first.invoice then do;
sum=0;
shipping=0;
foundcustomer = customer.find();
if foundcustomer eq 0 then do;
  obj.open_dir(name: "Invoice",
               label: "Invoice (Number=" || strip(invoice) ||
                    ") for customer " || strip(name),
               by: 1);

```

Display customer address information, an invoice number table, and heading information for the products purchased.

```

obj.layout_gridded(columns: 2);
obj.region(width: "4.5in");
  obj.format_text(data: strip(name),
                 overrides: "font_size=14pt width=100pct just=left");
  obj.format_text(data: strip(street),
                 overrides: "font_size=14pt width=100pct just=left");
  obj.format_text(data: strip(city) || ", " || state || " " || zip,
                 overrides: "font_size= 14pt width=100pct just=left" );
  obj.format_text(data: " ");
  obj.format_text(data: put(date, worddate18.),
                 overrides: "font_size=14pt width=100pct just=left");

obj.region();
obj.table_start();
  obj.row_start();
  obj.format_cell(data: "Invoice Number:",
                 overrides: "font_size=16pt font_weight=bold");
  obj.format_cell(data: invoice,
                 overrides: "width=1.5in font_size=16pt");
  obj.row_end();
obj.table_end();
obj.layout_end();

obj.table_start(overrides: "width=100pct");
obj.head_start();
  obj.row_start();
  obj.format_cell(data: "Items(s)",
                 column_span: 2);
  obj.format_cell(data: "Quantity");
  obj.format_cell(data: "Regular Price");
  obj.format_cell(data: "Sales Price");
  obj.format_cell(data: "Total");
  obj.row_end();
obj.head_end();
end;
end;

```

For every product purchased, add a row to the table describing the item, showing the catalog photo, quantity purchased, price information, and total cost.

```

if ( foundcustomer eq 0 ) and (product.find() eq 0 ) then do;
  if saleprice ne 0 then
    cost = quantity*saleprice;
  else
    cost = quantity*regprice;
sum = sum + cost;
shipping = shipping + additionalshipping;

obj.row_start();
  obj.format_cell(data: item,

```

```

                column_span:2,
                inhibit: "B",
                overrides: "just=left");
obj.format_cell(data: quantity,
                inhibit: "B");
obj.format_cell(data: regprice,
                inhibit: "B");
obj.format_cell(data: saleprice,
                inhibit: "B");
obj.format_cell(data: put(cost, dollar10.2),
                inhibit: "B");
obj.row_end();
obj.row_start();
obj.format_cell(data: " ",
                column_span: 2,
                overrides: "preimage='c:\public\Images\' || photo || ' just=left' );
obj.format_cell(data: " ");
obj.format_cell(data: " ");
obj.format_cell(data: " ");
obj.format_cell(data: " ");
obj.row_end();
end;

```

When we have reached the final observation for this invoice, display the Subtotal information, additional shipping expense, tax, and grand total.

```

if last.invoice then do;
obj.row_start();
obj.format_cell(data: " ",
                column_span: 3,
                row_span: 4 );
obj.format_cell(data: "Subtotal",
                column_span: 2,
                overrides: "just=right color=darkgray");
obj.format_cell(data: put(sum, dollar10.2));
obj.row_end();
obj.row_start();
obj.format_cell(data: "US Shipping",
                column_span: 2,
                overrides: "just=right color=darkgray",
                inhibit: "TB");
obj.format_cell(data: put(shipping, dollar10.2));
obj.row_end();
obj.row_start();
obj.format_cell(data: "Tax",
                column_span: 2,
                overrides: "just=right color=darkgray",
                inhibit: "TB");
tax = sum*.08;
obj.format_cell(data: put(tax, dollar10.2));
obj.row_end();

obj.row_start();
obj.format_cell(data: "Total",
                column_span: 2,
                overrides: "just=right font_weight=bold");
obj.format_cell(data: put(sum+tax+shipping, dollar10.2));
obj.row_end();
obj.table_end();
obj.close_dir();
end;

< more ... >

;run;

```


CONCLUSION

The ODS Report Writing Interface extends SAS report writing capabilities to address custom report writing capabilities. It is fully integrated with ODS features such as proportional fonts, traffic lighting, using colors, images, and unicode characters. The interface provides pixel-perfect placement capabilities while at the same time taking advantage of the rich programming features that the DATA step offers. With the flexibility of the DATA step and enhance ODS reporting features available, even the most rigid reporting requirements can be met with ease.

ACKNOWLEDGMENTS

The author would like to thank Allison Crutchfield for her contributions to this paper.

REFERENCES

“Getting Started with the DATA Step Hash Object” by Jason Secosky and Janice Bloom; 2007; SAS Institute Inc.

[Http://support.sas.com/rnd/base/datastep/hash-getting-started.pdf](http://support.sas.com/rnd/base/datastep/hash-getting-started.pdf)

CONTACT INFORMATION

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

Daniel O'Connor
SAS Institute Inc.
Building R, SAS Campus Drive
Cary, NC 27513
919-531-6171
Dan.OConnor@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.

APPENDIX 1 EXAMPLES

Example 1.1 Using TEXT Methods

```
ods listing close;
options nodate nonumber;
ods escapechar="~";
options papersize=(6in 4in);

data brand;
length tagline $24 description $128;
input tagline $24. / description $128.;
datalines;
Our Mission
To deliver the best quality sporting equipment, accessories, and outdoor equipment for
all seasons at the most affordable prices.
Our Vision
To transform the way the world purchases sporting and outdoor equipment.
Our Values
Customer focused, Swift and Agile, Innovative, Trustworthy
Our Goal
To grow sales by 15% annually while also improving profit margin through innovative
thinking and operational efficiencies.
;

title "~{style [preimage='c:\images\orionstarHeader.jpg' width=100pt
background=cx494068 color=cxbbb2e0 font_size=24pt just=left] Our Company}";
footnote "~{style [font_size=10pt just=right color=cxbbb2e0]Provided to you
compliments of SAS 9.2 using ODS Report Writing Interface using Text methods.}";
ods pdf file="text.pdf" notoc;

data _null_;
set brand end=eof;
if _n_ =1 then do;
  declare odsout obj();
  obj.note(data: "Our Brand...",
    overrides: "preimage='c:\images\star.gif'
                font_style=italic
                font_size=12pt
                font_weight=bold
                color=cxbbb2e0",
    just: "L");
end;
obj.format_text(data: tagline,
  overrides: "background=cx494068
              color=cxbbb2e0
              font_size=12pt
              font_style=italic
              width=2.5in",
  just: "C");
obj.format_text(data: description,
  overrides: "background=cxbbb2e0
              font_style=italic
              font_size=8pt
              width=2.5in",
  just: "C");
if eof ne 1 then
  obj.format_text(data: " ",
    overrides: "height=1mm");
;run;
ods _all_ close;
```

Example 1.2 Using Table Methods

```
ods listing close;
options nodate nonumber;
```

```

ods escapechar="~";
options papersize=(6in 4in);
title "~{style [preimage='c:\images\orionstarHeader.jpg' width=100pt
background=cx494068 color=cxbbb2e0 font_size=32pt just=left]}";
footnote "~{style [font_size=10pt just=right color=cxbbb2e0]Provided to you
compliments of SAS 9.2 using ODS Report Writing Interface using Table methods.}";
ods pdf file="stockquote.pdf";

data _null_;
dcl odsout obj();
obj.table_start(name: "Stock",
                label: "Stock Quote",
                overrides: "width=4in");
obj.row_start();
obj.format_cell(data: "STAR (Common Stock)",
                column_span: 2,
                overrides: "backgroundcolor=cx494068 color=white");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Exchange",
                overrides: "just=left font_weight=bold");
obj.format_cell(data: "NYSE (US Dollar)",
                overrides: "just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Price",
                overrides: "just=left font_weight=bold");
obj.format_cell(data: "$23.54",
                overrides: "just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Change (%)",
                overrides: "just=left font_weight=bold");
obj.format_cell(data: "0.12 (.51%)",
                overrides: "just=left color=green");
obj.row_end();
obj.row_start();
obj.cell_start();
obj.format_text(data: "Volume",
                overrides: "just=left font_weight=bold");
obj.cell_end();
obj.format_cell(data: "11,390",
                overrides: "just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Data as of 1/7/2009 9:43 a.m. ET *Minimum 20 minute delay",
                column_span: 2,
                split: '*',
                overrides: "font_size=10pt just=left");
obj.row_end();
obj.table_end();
run;

ods _all_ close;

```

Example 1.3 Using Gridded Layout Methods

```

ods listing close;
options nodate nonumber;
ods escapechar="~";
options papersize=(6in 4in);
data execs;
length photo $64 name $32 bio $256;
input photo $64. / name $32. bio $256.;
datalines;
c:\images\scott.jpg
Scott Huntley

```

Scott Huntley is the CEO and President of Orion Star, which he co-founded in 1976. Orion leads the industry in innovation with its award-winning recreational and outdoor equipment.

c:\images\dan.jpg

Dan OConnor

Dan OConnor is the CIO of Orion Star, which he also helped co-found in 1976. Orion continues to revolutionize the sporting goods and outdoors accessory industry through the adoption of technology to offer the best products at the most affordable price.

```
;
title "~{style [preimage='c:\images\orionstarHeader.jpg' width=100pct
background=cx494068 color=cxbbb2e0 font_size=32pt just=left] Our Founders}";
footnote "~{style [font_size=10pt just=right color=cxbbb2e0]Provided to you
compliments of SAS 9.2 using ODS Report Writing Interface Gridded Layout features.}";

ods pdf file="founders.pdf" notoc;
data _null_;
set execs end=eof;
if _n_=1 then do;
    dcl odsout obj();
    obj.layout_gridded(columns: 2,
                      overrides: "just=center");
end;
obj.region(width: "1.25in");
obj.image(file: photo,
          overrides: "just=left width=100pct");
obj.format_text(data: strip(name),
                overrides: "just=left width=100pct");
obj.region(width: "3.25in");
obj.format_text(data: strip(bio));

if eof ne 1 then
    obj.format_text(data: "");
else
    obj.layout_end();
;run;

ods _all_ close;
```

Example 1.4 Using Absolute Layout Methods

```
title;
ods listing close;
options nodate nonumber;
ods escapechar="~";
options papersize=(6in 4in);
options topmargin=0in bottommargin=0in leftmargin=0in rightmargin=0in;

proc format;
value categoryfmt
    1 = "Hunting"
    2 = "Dog"
    3 = "Sports"
    4 = "Children"
    5 = "Fresh Water Fishing"
    6 = "Salt Water Fishing"
    7 = "Camping"
    other = " ";
data customers;
format preferredcategory categoryfmt.;
input name $20. email $24. street $22. city $12. state $3. zip preferredcategory;
cards;
Dan O'Connor          Dan.OConnor@sas.com          SAS Campus Drive          Cary          NC
27513 0
```

```

Scott Huntley      Scott.Huntley@sas.com  105 Windward Way      Raleigh      NC
27615 3
David Kelley      David.Kelley@sas.com   507 Down Patrick Lane Garner      NC
27644 5
Wayne Hester      Wayne.Hester@sas.com   201 Gucci Boulevard   Cary      NC
27513 7
Tim Hunter        Tim.Hunter@sas.com     95 Wild Ranch Road    Taos      NM
89875 6
Eric Gebhart      Eric.Gebhart@sas.com   99 Sea Scape Island    Charleston SC
83478 2
Darylene Hecht    Darylene.Hecht@sas.com 300 Vintage Drive      Sonoma      CA
67676 4
Kevin Smith       Kevin.Smith@sas.com    28901 Pop Circle       Miami      FL
30497 1

```

```

;
run;

```

```

proc sort data=customers; by name;run;

```

```

*title;
*footnote "~{style [font_size=9pt just=right color=cxbbb2e0]Provided to you
compliments of SAS 9.2 using ODS Report Writing Interface and Absolute Layout
methods.}";
ods pdf file="Promotion.pdf" notoc newfile=Bygroup;

data _null_;
set customers end=eof;
by name;
if _n_ =1 then do;
  declare odsout obj();
end;

obj.open_dir(name: name,
             label: email,
             by: 1);

obj.layout_absolute(overrides: "borderwidth=1");
obj.region(width: "2.4in",
           height: "4in",
           overrides: "background=cx494068");
obj.image(file: "c:\Public\SGF 2009\Orion Star
Info\or_internal\orionstarHeader.jpg");
obj.format_text(data: "Private Sale",
               overrides: "color=cxbbb2e0 font_size=32pt",
               just: 'c');
obj.layout_gridded(columns: 3,
                  row_gutter: "1mm",
                  column_gutter: "1mm");

obj.region();
obj.format_text(data: "25",
               overrides: "color=cxbbb2e0 font_size=72pt");

obj.region();
obj.format_text(data: "%*off",
               split: "**",
               overrides: "color=cxbbb2e0 font_size=32pt");

obj.region();
obj.format_text(data: "Now *through *March 25",
               split: "**",
               overrides: "color=cxbbb2e0 font_size=10pt");

obj.layout_end();
obj.format_text(data: "your entire purchase *when you bring this card",
               split: "**",
               overrides: "color=cxbbb2e0 font_size=12pt width=85pct",
               just: 'C');
obj.format_text(data: "Excludes previously discounted items, layaways and prior
sales. Cannot be combined with any other offer.",

```

```

        overrides: "color=cxbbb2e0 font_size=4pt width=85pct",
        just: 'c');
obj.format_text(data: " ");
obj.format_text(data: "No payments for 12 months*",
    overrides: "width=100pct font_size=10pt background=white
color=cx494068 just=center");
obj.format_text(data: "*on purchases over $500 with your Orion Start preferred
Account. The Orion Star Preferred Account is subject to credit approval as determined
by the lender Outdoors Bank, Burlington, Vermont, is available to qualified US
residents, and is governed by Vermont and Federal Law. The promotion End Date for
this offer is the last day of the calendar month which is twelve months from the date
of purchase. If balance is not paid in full by the Promotion End Date show on your
billing statement, Finance Charges will accrue from the transaction date at an ANNUAL
PERCENTAGE RATE of 21.99%. A minimum Finance Charge of $2.00 will apply. Offer valid
in Orion Star stores only and expires 3/31/2009.",
    overrides: "color=cxbbb2e0 font_size=4pt width=85pct",
    just: 'c');

obj.region(x: "2.4in",
    width: "1.25in",
    height: "4in",
    overrides: "background=cx494068 borderwidth=1");
obj.table_start(overrides: "rules=none frame=void cellpadding=0 cellspacing=0");
obj.row_start();
obj.format_cell(data: "Pleasant Valley Promenade",
    overrides: "color=cxbbb2e0 font_size=6pt font_weight=bold");
obj.row_end();
obj.row_start();
obj.format_cell(data: "6204-121 Glennwood ave.",
    overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Raleigh, NC",
    overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "919.432.0987",
    overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "", overrides:"font_size=6pt");
obj.row_end();

obj.row_start();
obj.format_cell(data: "The streets at Southpoint",
    overrides: "color=cxbbb2e0 font_size=6pt font_weight=bold");
obj.row_end();
obj.row_start();
obj.format_cell(data: "3454 Executor Way",
    overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Durham, NC",
    overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "919.320.6238",
    overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Crabtree Valley Mall",
    overrides: "color=cxbbb2e0 just=left font_size=6pt font_weight=bold");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Raleigh, NC",

```

```

        overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "919.432.0987",
        overrides: "color=white font_size=6pt just=left");
obj.row_end();

obj.row_start();
obj.format_cell(data: "");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Store Hours",
        overrides: "color=cxbbb2e0 font_size=6pt just=left font_weight=bold");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Monday - Saturday",
        overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "10am - 8pm",
        overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Sunday",
        overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Noon - 5pm",
        overrides: "color=white font_size=6pt just=left");
obj.row_end();
obj.row_start();
obj.format_cell(data: "");
obj.row_end();
obj.table_end();

obj.table_start(overrides: "frame=box");
obj.row_start();
obj.format_cell(data: "Free Gift Wrapping",
        overrides: "color=cxbbb2e0 font_size=10pt just=center vjust=middle
width=1in height=.5in font_weight=bold");
obj.row_end();
obj.table_end();

obj.region(x: "1.9in",
        width: "2.75in",
        height: "4in");
obj.format_text(data: "Exclusive *Invitation *for our *Club *Members",
        split: "*",
        overrides: "color=cx494068 font_size=36pt");
obj.format_text(data: " ");

obj.format_text(data: name);
obj.format_text(data: street);
obj.format_text(data: strip(city) || ", " || state || " " || zip);

obj.layout_end();

obj.close_dir();

;run;

ods pdf close;

title;
footnote;

```

Example 1.5 Creating Form Letters

```
ods listing close;
options nodate nonumber stimer fullstimer;
ods escapechar="~";
options papersize=letter;
options topmargin=0in bottommargin=0in leftmargin=0in rightmargin=0in;

proc format;
value prodstate
  1 = "Back Order"
  2 = "Special Order"
  other = "Regular Inventory";
value categoryfmt
  1 = "Hunting"
  2 = "Dog"
  3 = "Sports"
  4 = "Children"
  5 = "Fresh Water Fishing"
  6 = "Salt Water Fishing"
  7 = "Camping"
  other = " ";
run;

data customers;
format preferredcategory categoryfmt.;
input name $20. email $24. street $22. city $12. state $3. zip preferredcategory
accountid clubpoints;
cards;
Dan O'Connor      Dan.OConnor@sas.com      SAS Campus Drive      Cary      NC 27513
0 1234567      256
Scott Huntley      Scott.Huntley@sas.com      105 Windward Way      Raleigh      NC
27615 3 9857954      0
David Kelley      David.Kelley@sas.com      507 Down Patrick Lane      Garner      NC
27644 5 4382743      1256
Wayne Hester      Wayne.Hester@sas.com      201 Gucci Boulevard      Cary      NC
27513 7 3485944      653
Tim Hunter      Tim.Hunter@sas.com      95 Wild Ranch Road      Taos      NM
89875 6 3294746      77
Eric Gebhart      Eric.Gebhart@sas.com      99 Sea Scape Island      Charleston SC
83478 2 9087450      943
Darylne Hecht      Darylne.Hecht@sas.com      300 Vintage Drive      Sonoma      CA
67676 4 4987654      10503
Kevin Smith      Kevin.Smith@sas.com      28901 Pop Circle      Miami      FL
30497 1 8734902      444
;
run;

data inventory;
format productstatus prodstate.;
input item $64. photo $24. itemno instock productstatus regprice saleprice
additionalshipping;
datalines;
Gorilla King Kong Hang-On Tree stand      treestand.jpg
89783483      34      0      89.99      79.99      12.00
Orion Star XT Mega-Sized Binoculars      binoculars.jpg
87634893      134      0      279.99      0.00      0.00
Night Owl Night Scope      nightowl.jpg
37274307      0      1      199.99      0.00      0.00
Super Doggy Yapper Stopper      trainingcollar.jpg
87393444      222      2      79.99      0.00      0.00
Neoprene Flotation Vest      vest.jpg
77439843      66      0      34.99      0.00      0.00
Dead Duck Retriever Trainers      mallard.jpg
33348484      256      0      44.55      0.00      0.00
```



```

Spilding Wood Baseball Bat                                baseball_bat.jpg
98457897 934 0 34.99 29.99 0.00
Foam Football                                            football.jpg
59498478 342 0 12.99 0.00 0.00
Super Start First Base Glove                            baseballglove.jpg
98754598 232 0 42.50 0.00 0.00
Little Princess Water proof Radio                       radios.jpg
54987430 5 0 25.99 0.00 0.00
Big Bubba Sonar                                         sonar.jpg
98745878 55 2 549.99 0.00 9.00
;

```

```

data transactions;
format date mmdyy10.;
input invoice accountid itemno quantity date:mmdyy10. clubvisa;
datalines;

```

```

1 1234567 89783483 1 01/02/2009 1
1 1234567 87634893 1 01/02/2009 1
2 1234567 37274307 1 01/05/2009 0
3 3294746 98745878 1 01/25/2009 0
4 9857954 59498478 3 01/23/2009 0
5 9857954 98457897 1 01/30/2009 0
5 9857954 98754598 6 01/30/2009 0
6 4987654 54987430 4 01/04/2009 1
7 9087450 87393444 1 01/14/2009 1
8 3485944 33348484 2 01/14/2009 0
9 3485944 77439843 1 01/26/2009 0

```

```
;run;
```

```
proc sort data=transactions; by accountid invoice;run;
```

```

title "~{style [preimage='c:\Public\SGF 2009\Orion Star
Info\or_internal\orionstarHeader.jpg' width=100pct background=cx494068 color=cxbbb2e0
font_size=24pt just=left] Preferred Club Member}";
footnote "~{style [font_size=9pt just=right color=cxbbb2e0]Provided to you compliments
of SAS 9.2 using ODS Report Writing Interface methods.}";
ods pdf file="Example1.5.pdf" notoc newfile=Bygroup;

```

```

data _null_;
retain invsum bonus sum clubpoints;
if _n_ =1 then do;
declare odsout obj();
length name $20 street $22 city $12 state $3;
zip=.;
declare hash customer(dataset: "customers");
customer.defineKey("accountid");
customer.defineData("name", "street", "city", "state", "zip", "clubpoints");
customer.defineDone();

length item $64;
itemno=.; instock=.; productstatus=.; regprice=.; saleprice=.; additionalshipping=.;
declare hash product(dataset: "inventory");
product.defineKey("itemno");
product.defineData("item", "instock", "productstatus", "regprice", "saleprice",
"additionalshipping");
product.defineDone();
end;

```

```

set transactions end=eof;
by accountid invoice;
if first.accountid then do;
sum=0;
obj.open_dir(name: "Invoice",
label: "Invoice for Customer " || accountid,
by: 1);

if customer.find() eq 0 then do;

```

```

obj.layout_gridded(columns: 2);
obj.region(width: "3.25in");
obj.format_text(data: strip(street), overrides: "font_size=14pt width=100pct
just=left");
obj.format_text(data: strip(city) || ", " || state || " " || zip, overrides:
"font_size= 14pt width=100pct just=left" );
obj.format_text(data: " ");
obj.format_text(data: put(today(), worddate18.), overrides: "font_size=14pt
width=100pct just=left");
obj.format_text(data: " ");
obj.format_text(data: "Dear " || strip(name) || ",", overrides:
"font_size=14pt width=100pct just=left");
obj.region();
obj.image(file: "c:\public\Images\Scenic2.jpg");
obj.layout_end();

obj.layout_gridded(columns: 2);
obj.region(width: "5.25in");
obj.format_text(data: "~{style [font_size=14pt width=100pct]As a preferred
Club Member you receive special membership benefits such as special promotions, member
discounts, free 1 year warranty on all purchases, as bonus points that can be redeemed
for free items.}", just: "L");
obj.format_text(data: " ");
obj.layout_gridded(overrides: "background=cxbbb2e0");
obj.region(overrides: "background=_undef_");
obj.format_text(data: "See what being a preferred Club Member is all
about.",
                overrides: "font_size=20pt font_weight=bold
width=100pct");
obj.format_text(data: "You are pre-approved for our new ~{style
[font_weight=bold]Orion Star Club Visa} card.",
                overrides: "just=left font_size=14pt width=100pct");
obj.format_text(data: " ");
obj.format_text(data: "1. A low 8.9%APR",
                overrides: "just=left font_size=12pt width=100pct");
obj.format_text(data: "2. Get double membership points when using your
Orion Star Club Visa card.",
                overrides: "just=left font_size=12pt width=100pct");
obj.format_text(data: "3. Get Exclusive access to offers not available to
general public",
                overrides: "just=left font_size=12pt width=100pct");
obj.format_text(data: " ");
obj.href(data: "~{style [font_size=20pt]Apply Today}",
         href: "http:\\www.orionstar.com\\visaapplication");

obj.layout_end();
obj.region(width: "2.5in");
obj.format_text(data: "Enjoy your Preferred membership",
                overrides: "color=cx494068 font_size=16pt font_weight=bold
width=100pct",
                just: "L");
obj.format_text(data: " ");
obj.format_text(data: "Special Promotions",
                overrides: "color=cx494068 font_size=14pt width=100pct",
                just: "L");
obj.format_text(data: " ");
obj.format_text(data: "Membership Discounts",
                overrides: "color=cx494068 font_size=14pt width=100pct",
                just: "L");
obj.format_text(data: " ");
obj.format_text(data: "FREE 1 year warranty",
                overrides: "color=cx494068 font_size=14pt width=100pct",
                just: "L");
obj.format_text(data: " ");
obj.format_text(data: "FREE Items",
                overrides: "color=cx494068 font_size=14pt width=100pct",
                just: "L");
obj.format_text(data: " ");

```

```

obj.layout_end();

obj.format_text(data: " ");

obj.table_start(overrides: "width=100pct");
obj.row_start();
obj.format_cell(data: "Current Club Point Balance",
                column_span: 2,
                overrides: "font_weight=bold font_size=16pt");
obj.format_cell(data: clubpoints,
                format: "comma8",
                overrides: "font_weight=bold font_size=16pt");
obj.row_end();
obj.row_start();
obj.format_cell(data: "Recent Purchases",
                column_span: 3 );
obj.row_end();
obj.row_start();
obj.format_cell(data: "Date");
obj.format_cell(data: "Invoice Number");
obj.format_cell(data: "Additional Club Points");
obj.row_end();
end;
end;

if first.invoice then do;
    invsum=0;
    bonus=0;
end;

if product.find() eq 0 then do;
    if saleprice ne 0 then
        cost = quantity*saleprice;
    else
        cost = quantity*regprice;
    end;
    invsum = invsum + cost;
    sum = sum + cost;
end;

if clubvisa eq 1 then do;
    bonus = bonus + clubvisa*cost;
    sum = sum + clubvisa*cost;
end;

if last.invoice then do;
    obj.row_start();
    obj.format_cell(data: date,
                    format: "mddyy10.");
    obj.format_cell(data: invoice);
    obj.format_cell(data: invsum,
                    format: "comma8.");
obj.row_end();

    if clubvisa eq 1 then do;
        obj.row_start();
        obj.format_cell(data: "Bonus Visa Club Points",
                        column_span: 2);
        obj.format_cell(data: bonus,
                        format: "comma8.");
        obj.row_end();
    end;
end;

if last.accountid then do;
    obj.row_start();
    obj.format_cell(data: "New Club Point Balance",
                    column_span: 2,
                    overrides: "font_weight=bold font_size=16pt");

```

```

    obj.format_cell(data: clubpoints+sum,
                    overrides: "font_weight=bold font_size=16pt",
                    format: "comma8.");
obj.row_end();

obj.table_end();

obj.format_text(data: " ",
                overrides: "font_size=16pt");
obj.format_text(data: "How to Redeem Points?",
                overrides: "background=cxbbb2e0 font_size=20pt");
obj.format_text(data: "Step-by-step instruction on how to use your Club REWARD Point
to purchase free items.",
                overrides: "font_size=16pt");
obj.format_text(data: " ");
obj.format_text(data: "~{style [font_size=20pt
url='http:\\www.orionstar\\redeempoints']Learn How!}");

obj.close_dir();
end;

if end eq 1 then do;
    obj.delete();
    customer.delete();
    product.delete();
end;
;run;

ods pdf close;
title;
footnote;

```

Example 1.6 Creating an Invoice

```

title;
ods listing close;
options nodate nonumber stimer fullstimer;
ods escapechar="~";
options papersize=letter;
options topmargin=0in bottommargin=0in leftmargin=0in rightmargin=0in;

proc format;
value prodstate
  1 = "Back Order"
  2 = "Special Order"
  other = "Regualr Inveentory";
value categoryfmt
  1 = "Hunting"
  2 = "Dog"
  3 = "Sports"
  4 = "Children"
  5 = "Fresh Water Fishing"
  6 = "Salt Water Fishing"
  7 = "Camping"
  other = " ";
run;

data customers;
format preferredcategory categoryfmt.;
input name $20. email $24. street $22. city $12. state $3. zip preferredcategory
accountid clubpoints;
cards;
Dan O'Connor          Dan.OConnor@sas.com      SAS Campus Drive      Cary      NC 27513
0 1234567      256
Scott Huntley         Scott.Huntley@.sas.com   105 Windward Way     Raleigh   NC
27615 3 9857954      0
David Kelley          David.Kelley@sas.com     507 Down Patrick Lane Garner     NC
27644 5 4382743      1256
Wayne Hester          Wayne.Hester@sas.com     201 Gucci Boulevard  Cary      NC
27513 7 3485944      653
Tim Hunter            Tim.Hunter@sas.com       95 Wild Ranch Road   Taos      NM
89875 6 3294746      77
Eric Gebhart          Eric.Gebhart@sas.com     99 Sea Scape Island   Charleston SC
83478 2 9087450      943
Darylne Hecht         Darylne.Hecht@sas.com   300 Vintage Drive     Sonoma    CA
67676 4 4987654      10503
Kevin Smith           Kevin.Smith@sas.com      28901 Pop Circle     Miami     FL
30497 1 8734902      444
;
run;

data inventory;
format productstatus prodstate.;
input item $64. photo $24. itemno instock productstatus regprice saleprice
additionalshipping;
datalines;
Gorilla King Kong Hang-On Tree stand          treestand.jpg
89783483 34 0 89.99 79.99 12.00
Orion Star XT Mega-Sized Binoculars          binoculars.jpg
87634893 134 0 279.99 0.00 0.00
Night Owl Night Scope                        nightowl.jpg
37274307 0 1 199.99 0.00 0.00
Super Doggy Yapper Stopper                  trainingcollar.jpg
87393444 222 2 79.99 0.00 0.00
Neoprene Flotation Vest                     vest.jpg
77439843 66 0 34.99 0.00 0.00

```

```

Dead Duck Retriever Trainers                                mallard.jpg
33348484 256 0 44.55 0.00 0.00
Spilding Wood Baseball Bat                                baseball_bat.jpg
98457897 934 0 34.99 29.99 0.00
Foam Football                                              football.jpg
59498478 342 0 12.99 0.00 0.00
Super Start First Base Glove                              baseballglove.jpg
98754598 232 0 42.50 0.00 0.00
Little Pricess Water proof Radio                          radios.jpg
54987430 5 0 25.99 0.00 0.00
Big Bubba Sonar                                           sonar.jpg
98745878 55 2 549.99 0.00 9.00
;

```

```

data transactions;
format date mmdyy10.;
input invoice accountid itemno quantity date:mmdyy10. clubvisa;
datalines;

```

```

1 1234567 89783483 1 01/02/2009 1
1 1234567 87634893 1 01/02/2009 1
2 1234567 37274307 1 01/05/2009 0
3 3294746 98745878 1 01/25/2009 0
4 9857954 59498478 3 01/23/2009 0
5 9857954 98457897 1 01/30/2009 0
5 9857954 98754598 6 01/30/2009 0
6 4987654 54987430 4 01/04/2009 1
7 9087450 87393444 1 01/14/2009 1
8 3485944 33348484 2 01/14/2009 0
9 3485944 77439843 1 01/26/2009 0

```

```
;run;
```

```
proc sort data=transactions; by invoice;run;
```

```

title "~{style [preimage='c:\Public\SGF 2009\Orion Star
Info\or_internal\orionstarHeader.jpg' width=100pt background=cx494068 color=cxbbb2e0
font_size=24pt just=left] Invoice}";
footnote "~{style [font_size=9pt just=right color=cxbbb2e0]Provided to you compliments
of SAS 9.2 using ODS Report Writing Interface methods.}";
ods pdf file="Example1.6.pdf" notoc newfile=Bygroup;

```

```

data _null_;
retain foundcustomer sum shipping;
if _n_ =1 then do;
declare odsout obj();
length name $20 street $22 city $12 state $3;
zip=.;
declare hash customer(dataset: "customers");
customer.defineKey("accountid");
customer.defineData("name", "street", "city", "state", "zip");
customer.defineDone();

length item $64 photo$24;
itemno=.; instock=.; productstatus=.; regprice=.; saleprice=.; additionalshipping=.;
declare hash product(dataset: "inventory");
product.defineKey("itemno");
product.defineData("item", "instock", "productstatus", "regprice", "saleprice",
"additionalshipping", "photo");
product.defineDone();
end;

set transactions end=eof;
by invoice;
if first.invoice then do;
sum=0;
shipping=0;
foundcustomer = customer.find();
if foundcustomer eq 0 then do;

```

```

    obj.open_dir(name: "Invoice",
                label: "Invoice (Number=" || strip(invoice) || ") for customer " ||
strip(name),
                by: 1);
    obj.layout_gridded(columns: 2);
    obj.region(width: "4.5in", overrides: "just=left");
    obj.format_text(data: strip(name), overrides: "font_size=14pt width=100pct
just=left");
    obj.format_text(data: strip(street), overrides: "font_size=14pt width=100pct
just=left");
    obj.format_text(data: strip(city) || ", " || state || " " || zip, overrides:
"font_size= 14pt width=100pct just=left" );
    obj.format_text(data: " ");
    obj.format_text(data: put(date, worddate18.), overrides: "font_size=14pt
width=100pct just=left");

    obj.region();
    obj.table_start();
    obj.row_start();
    obj.format_cell(data: "Invoice Number:",
                    overrides: "font_size=16pt font_weight=bold");
    obj.format_cell(data: invoice,
                    overrides: "width=1.5in font_size=16pt");
    obj.row_end();
    obj.table_end();
    obj.layout_end();

    obj.format_text(data: " ");
    obj.format_text(data: " ");

    obj.table_start(overrides: "width=100pct");
    obj.head_start();
    obj.row_start();
    obj.format_cell(data: "Items(s)",
                    column_span: 2);
    obj.format_cell(data: "Quantity");
    obj.format_cell(data: "Regular Price");
    obj.format_cell(data: "Sales Price");
    obj.format_cell(data: "Total");
    obj.row_end();
    obj.head_end();
end;
end;

if ( foundcustomer eq 0 ) and (product.find() eq 0 ) then do;
if saleprice ne 0 then
    cost = quantity*saleprice;
else
    cost = quantity*regprice;
sum = sum + cost;
shipping = shipping + additionalshipping;

obj.row_start();
    obj.format_cell(data: item,
                    column_span:2,
                    inhibit: "B",
                    overrides: "just=left");
    obj.format_cell(data: quantity,
                    inhibit: "B");
    obj.format_cell(data: regprice,
                    inhibit: "B");
    obj.format_cell(data: saleprice,
                    inhibit: "B");
    obj.format_cell(data: put(cost, dollar10.2),
                    inhibit: "B");
obj.row_end();
obj.row_start();

```

```

        obj.format_cell(data: " ",
                        column_span: 2,
                        overrides: "preimage='c:\public\Images\" || photo || "
just=left" );
        obj.format_cell(data: " ");
        obj.format_cell(data: " ");
        obj.format_cell(data: " ");
        obj.format_cell(data: " ");
        obj.row_end();
end;

if last.invoice then do;
    obj.row_start();
    obj.format_cell(data: " ",
                    column_span: 3,
                    row_span: 4 );
    obj.format_cell(data: "Subtotal",
                    column_span: 2,
                    overrides: "just=right color=darkgray");
    obj.format_cell(data: put(sum, dollar10.2));
    obj.row_end();
    obj.row_start();
    obj.format_cell(data: "US Shipping",
                    column_span: 2,
                    overrides: "just=right color=darkgray",
                    inhibit: "TB");
    obj.format_cell(data: put(shipping, dollar10.2));
    obj.row_end();
    obj.row_start();
    tax = sum*.08;
    obj.format_cell(data: "Tax",
                    column_span: 2,
                    overrides: "just=right color=darkgray",
                    inhibit: "TB");
    obj.format_cell(data: put(tax, dollar10.2));
    obj.row_end();
    obj.row_start();
    obj.format_cell(data: "Total",
                    column_span: 2,
                    overrides: "just=right font_weight=bold");
    obj.format_cell(data: put(sum+tax+shipping, dollar10.2));
    obj.row_end();
    obj.table_end();
    obj.close_dir();
end;

if end eq 1 then do;
    obj.delete();
    customer.delete();
    product.delete();
end;
;run;

ods pdf close;
title;
footnote;

```


TEXT METHODS

FORMAT_TEXT**Syntax**

Object.format_text(< optional argument >, ... , < optional argument >);

Description

Displays text in the active output destination(s).

Optional Arguments

data	< string number character variable numeric variable >
	The data value to display. If the data is numeric and no format has been specified, the data value will be formatted using the BEST. format.
format	< string character variable >
	The SAS format to be applied to the data argument.
style	< string character variable >
	The style element that contains the collection of style attributes to be applied to the data value. The default style element is USERTEXT.
overrides	< string character variable >
	The style attributes to override those defined in the selected style element.
split	< string character variable >
	Split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not considered part of the data value.
no_clean	< number numeric variable >
just	< single character single character variable >
	Horizontal justification for the data value.
	Valid values
	L Left justification
	C Center justification
	R Right justification
	D Decimal point justification
vjust	< single character single character variable >
	Vertical justification for the data value.
	Valid values
	T Top justification
	M Middle justification
	B Bottom justification

Example

```
obj.format_text(data: "Display this text in the active output
destination.");

obj.format_text(data: "Make this text look like The SAS System
title.",
                style: "TitlesandFooters");

obj.format_text(data: "Make this text red",
                overrides: "color=red",
                data: "and this bold",
                overrides: "font_weight=bold",
                data: " and use a 16pt font",
                overrides: "font_size=16pt");
```

NOTE

Syntax

Object.note(< optional argument >, ... , < optional argument >);

Description

Writes a note to the active output destination(s).

Optional Arguments

data	< string number character variable numeric variable >
	The data value to display. If the data is numeric and no format has been specified the data value will be formatted using the BEST. format.
format	< string character variable >
	The SAS format to be applied to the data argument.
style	< string character variable >
	The style element that contains the collection of style attributes to be applied to the data value. The default style element is "note".
overrides	< string character variable >
	The style attributes to override those defined in the selected style element.
split	< string character variable >
	Split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not considered part of the data value.
no_clean	< number numeric variable >
just	< single character single character variable >
	Horizontal justification for the data value.
	Valid values
	'L' Left justification
	'C' Center justification
	'R' Right justification
	'D' Decimal point justification
vjust	< single character single character variable >

Vertical justification for the data value.

Valid values

'T'	Top justification
'M'	Middle justification
'B'	Bottom justification

Example

```
obj.note(data: "This is the text to display in a note.");
```

TABLE SECTION METHODS

HEAD_START

Syntax

```
Object.head_start( < optional argument >, ... , < optional argument > );
```

Description

Marks the start of the Table Header Section. The HEAD_START is always used in conjunction with the HEAD_END method. You can also use the TYPE argument with the ROW_START method as an alternate approach.

Optional Arguments

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element is "Header".

overrides < string | character variable >

The style attributes to override those defined in the selected style element.

Example

```
obj.table_start();  
obj.head_start();  
obj.row_start();  
obj.format_cell(data: "A single cell table");  
obj.row_end();  
obj.head_end();  
obj.table_end();
```

HEAD_END

Syntax

```
Object.head_end( );
```

Description

Marks the end of the Table Header section. The HEAD_END is always used in conjunction with the HEAD_START method.

No Arguments

BODY_START

Syntax

```
Object.body_start( < optional argument >, ... , < optional argument > );
```

Description

Marks the start of the Table Body(Data) Section. The BODY_START is always used in conjunction with the BODY_END method. You can also use the "TYPE" argument with the ROW_START method as an alternate approach. This is the default section if you just use the ROW_START method.

Optional Arguments

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element is "Body".

overrides < string | character variable >

The style attributes to override those defined in the selected style element.

Example

```
obj.table_start();
obj.body_start();
obj.row_start();
obj.format_cell(data: "A single cell table");
obj.row_end();
obj.body_end();
obj.table_end();
```

BODY_END

Syntax

Object.body_end();

Description

Marks the end of the Table Body(Data) section. The BODY_END is always used in conjunction with the BODY_START method.

No Arguments

FOOT_START

Syntax

Object.foot_start(< optional argument >, ... , < optional argument >);

Description

Marks the start of the Table Footer Section. The FOOT_START is always used in conjunction with the FOOT_END method. You can also use the TYPE argument with the ROW_START method as an alternate approach.

Optional Arguments

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element is "Footer".

overrides < string | character variable >

The style attributes to override those defined in the selected style element.

Example

```
obj.table_start();
obj.foot_start();
obj.row_start();
```

```
        obj.format_cell(data: "A single cell table");
    obj.row_end();
    obj.foot_end();
obj.table_end();
```

FOOT_END**Syntax**

Object.foot_end();

Description

Marks the end of the Table Footer section. The FOOT_END is always used in conjunction with the FOOT_START method.

No Arguments

TABLE METHODS

TABLE_START

Syntax

Object.table_start(< optional argument >, ... , < optional argument >);

Description

The TABLE_START is always used in conjunction with the TABLE_END method.

Optional Arguments

name The name of the table that will be used in the table of contents (TOC) and DMS Results window.

label The label of the table that will be used in the table of contents (TOC) and DMS Results window.

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element is TABLE.

overrides < string | character variable >

The style attributes to override those defined in the selected style element.

just < single character | single character variable >

Horizontal justification for the data value.

Valid values

L Left justification
C Center justification
R Right justification
D Decimal point justification

vjust < single character | single character variable >

Vertical justification for the data value.

Valid values

T Top justification
M Middle justification
B Bottom justification

top_space < number | numeric variable >

The number of blank lines to insert before the table starts.

Example

```
obj.table_start();  
  obj.row_start();  
    obj.format_cell(data: "A single cell table");  
  obj.row_end();  
obj.table_end();
```

TABLE_END

Syntax

Object.table_end();

Description

The TABLE_END is always used in conjunction with the TABLE_START method.

No Arguments

Example

```
obj.table_start();
  obj.row_start();
    obj.format_cell(data: "A single cell table");
  obj.row_end();
obj.table_end();
```

ROW_START

Syntax

Object.row_start(< optional argument >, ... , < optional argument >);

Description

The ROW_END is always used in conjunction with the ROW_START method.

Optional Arguments

type Correspond to the table sections. This is just an alternative to having to use the HEAD_START and HEAD_END method calls. The default type is BODY.

Valid values

H	Header Section
D	Body(data) Section
B	Body(data) Section
F	Footer Section

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element depends on the row type.

overrides < string | character variable >

The style attributes to override those defined in the selected style element.

Row The row index. The table keeps track of its current row index so this allows you to skip blank rows. Once you have skipped a row, you cannot index back to a previous row.

Examples

```
obj.table_start();
  obj.row_start();
    obj.format_cell(data: "A single cell table");
  obj.row_end();
obj.table_end();
```

ROW_END

Syntax

Object.row_end();

Description

The ROW_END is always used in conjunction with the ROW_START method.

No Arguments

Example

```
obj.table_start();
  obj.row_start();
    obj.format_cell(data: "A single cell table");
  obj.row_end();
obj.table_end();
```

CELL_START

Syntax

Object.cell_start(< optional argument >, ... , < optional argument >);

Description

The CELL_END is always used in conjunction with the CELL_START method.

Optional Arguments

data	< string number character variable numeric variable >
	The data value to display. If the data is numeric and no format has been specified, the data value will be formatted using the BEST. format.
format	< string character variable >
	The SAS format to be applied to the data argument.
style	< string character variable >
	The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
overrides	< string character variable >
	The style attributes to override those defined in the selected style element.
split	< string character variable >
	Split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not considered part of the data value.
no_clean	< number numeric variable >
Column	< number numeric variable >
Column_span	< number numeric variable >
Row_span	< number numeric variable >
Inhibit	Tells aspects of the cells that may be inhibited. This option can be honored only for certain destinations; in particular, HTML does not currently support it.
	Valid values
T	Do not draw the top border. Note that some destinations will have already drawn a rule at the bottom of the previous row, so this one may not be effective.
B	Do not draw the bottom border of this cell.
L	Do not draw the left border. Ineffective if the destination already drew that rule on the right of the previous cell.
R	Do not draw the right border of this cell.

- X Do not draw the contents of the cell, just the background. Usually desirable on one of the two cells which are using the "B" or "R".

Example

```
obj.table_start();
  obj.row_start();
    obj.cell_start()
      obj.format_text(data: "A single cell table");
    obj.cell_end();
  obj.row_end();
obj.table_end();
```

CELL_END

Syntax

```
Object.cell_end();
```

Description

The CELL_END is always used in conjunction with the CELL_START method.

No Arguments

Example

```
obj.table_start();
  obj.row_start();
    obj.cell_start()
      obj.format_text(data: "A single cell table");
    obj.cell_end();
  obj.row_end();
obj.table_end();
```

FORMAT_CELL

Syntax

```
Object.format_cell( < optional argument >, ... , < optional argument > );
```

Description

Optional Arguments

- data** < string | number | character variable | numeric variable >
The data value to display. If the data is numeric and no format has been specified the data value will be formatted using the BEST. format.
- format** < string | character variable >
The SAS format to be applied to the data argument.
- style** < string | character variable >
The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
- overrides** < string | character variable >
The style attributes to override those defined in the selected style element.
- split** < string | character variable >
Split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not considered part of the data value.

no_clean	< number numeric variable >
Column	< number numeric variable >
Column_span	< number numeric variable >
Row_span	< number numeric variable >
Inhibit	Tells aspects of the cells that may be inhibited. This option can be honored only for certain destinations; in particular, HTML does not currently support it.
	Valid values
T	Do not draw the top border. Note that some destinations will have already drawn a rule at the bottom of the previous row, so this one may not be effective.
B	Do not draw the bottom border of this cell.
L	Do not draw the left border. Ineffective if the destination already drew that rule on the right of the previous cell.
R	Do not draw the right border of this cell.
X	Do not draw the contents of the cell, just the background. Usually desirable on one of the two cells which are using the "B or "R".

Example

```
obj.table_start();

/* Row 1 */
obj.row_start();
  obj.format_cell(data: "Cell 1");
  obj.format_cell(data: "Cell 2");
obj.row_end();

/* Row 2 */
obj.row_start();
  obj.format_cell(data: "Cell 1");
  obj.format_cell(data: "Cell 2");
obj.row_end();

obj.table_end();

obj.table_start();

/* Row 1 will span multiple cells */
obj.row_start();
  obj.format_cell(data: "This is a spanning cell",
                  column_span: 2,
                  style: "Header");
obj.row_end();

/* Row 2 has separate cells */
obj.row_start();
  obj.format_cell(data: "Cell 1");
  obj.format_cell(data: "Cell 2");
obj.row_end();
obj.table_end();
```

PAGE METHODS

PAGE

Syntax

```
Object.page( );
```

Description

Forces a page eject or configures the page.

No Arguments

Example

```
obj.page( );
```

TITLE

Syntax

```
Object.title(< optional argument >, ... , < optional argument >);
```

Description

Adds a new page title to the system.

Optional Arguments

text < string | character variable >

The text value to insert into the title system processing.

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element is TITLE.

Start The title index to begin at. Valid numeric range 1–10.

Clear Clear the cached title.

Example

```
obj.title(text: "Here is a new title");  
/* Clear "The SAS System" default title */  
obj.title(start: 1,  
         clear: 1);
```

FOOTNOTE

Syntax

```
Object.footer(< optional argument >, ... , < optional argument >);
```

Description

Adds a new page footnote to the system.

Optional Arguments

text < string | character variable >

The text value to insert into the footnote system processing.

style < string | character variable >

The style element that contains the collection of style attributes to be applied to the data value. The default style element is FOOTNOTE.

Start The footnote index to begin at. Valid numeric range 1–10.
Clear Clear the cached footnote.

Example

```
obj.footnote(text: "Here is a new footnote");  
/* Clears the previous footnote */  
obj.footnote(start: 1,  
             clear: 1);
```

GRIDDED LAYOUT METHODS

LAYOUT_GRIDDED

Syntax

Object.layout_gridded(< optional argument >, ... , < optional argument >);

Description

Creates a new gridded layout.

Optional Arguments

- x** < dimension unit >
Horizontal position of the LAYOUT, which will extend to the right of this position for WIDTH. If omitted, it defaults to 0.
- y** < dimension unit >
Vertical position of the LAYOUT, which will extend down from this position for HEIGHT. If omitted, it defaults to the current vertical position on the page.
- width** < dimension unit >
Horizontal width of the LAYOUT. If omitted, it defaults to the maximum horizontal space needed to display all regions.
- height** < dimension unit >
Vertical height of the LAYOUT. If omitted, it defaults to the maximum vertical space needed to display all regions.
- columns** < number | numeric variable >
Fixed number of columns in the gridded LAYOUT. If omitted, it defaults to 1 column.
- column_widths** < dimension unit >
Width of each column specified. This is a space-delimited list of horizontal sizes that correspond to each column. The number of horizontal sizes must match the number of columns specified, or else a warning will be produced, and the option will be ignored.
- column_gutter** < dimension unit >
Horizontal space between each column. If omitted, it defaults to the CELL_SPACING style attribute.

rows	< number numeric variable >	Fixed number of rows in the gridded LAYOUT. If omitted, it defaults to the maximum number of regions created in the vertical direction. This option should be used very sparingly.
row_heights	< dimension unit >	Height of each row specified. This is a space-delimited list of vertical sizes that correspond to each row. The number of vertical sizes must match the number of rows specified or else a warning will be produced, and the option will be ignored.
row_gutter	< dimension unit >	Vertical space between each row. If omitted, it defaults to the CELL_SPACING style attribute.
style	< string character variable >	The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
overrides	< string character variable >	The style attributes to override those defined in the selected style element.

Example:

```
obj.layout_gridded(columns: 1);
  obj.region();
    obj.format_text(data: "Here is a some text for a region.");
obj.layout_end();
```

REGION

Syntax

Object.region(< optional argument >, ... , < optional argument >);

Description

Create a region that will contain some output.

Optional Arguments

width	Horizontal width of the REGION, and is restricted by the LAYOUT dimensions. If omitted, it defaults to the maximum horizontal space needed to display the output contained in the REGION. The sum of all region widths cannot exceed the LAYOUT horizontal dimension.
height	Vertical height of the REGION, and is restricted by the LAYOUT dimensions. If omitted, it defaults to the maximum vertical spaced needed to display the output contained in the REGION. The sum of all region heights cannot exceed the LAYOUT vertical dimension. This option should be used very sparingly.
column	Allows you to specify the current grid column position in the gridded layout. This is generally useful only when you want to skip regions in the gridded layout and should be used very sparingly. The gridded layout automatically tracks the current grid column position and will be incremented for every region statement. Once you have skipped a grid column, you cannot go back to them. Random access of grid rows and columns is not supported.
column_span	Allows you to specify the number of grid columns that the region will occupy. It simply allows you to combine adjacent grid columns in

gridded layout. Default is 1.

row	Allows you to specify the current grid row position in the gridded layout. This is generally only useful when you want to skip regions in the gridded layout, and should be used very sparingly. The gridded layout automatically tracks the current row position, and will be incremented for every region statement. Once you have skipped rows, you cannot go back to them. Random access to row and columns is not supported.
row_span	Allows you to specify the number of grid rows that the region will occupy. It simply allows you to combine adjacent grid rows in gridded layout. Default is 1.
style	The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
overrides	The style attributes to override those defined in the selected style element.

Example:

```
obj.layout_gridded(columns: 1);  
  obj.region();  
    obj.format_text(data: "Here is a some text for a region.");  
obj.layout_end();
```

LAYOUT_END

Syntax

```
Object.layout_end( );
```

Description

Close the active layout.

No Arguments

Example

```
obj.layout_end( );
```

ABSOLUTE LAYOUT METHODS

LAYOUT_ABSOLUTE

Syntax

```
Object.layout_absolute(< optional argument >, ... , < optional argument >);
```

Description

Create an absolute layout container.

Optional Arguments

x < dimension unit >

Horizontal position of the LAYOUT, which will extend to the right of this position for WIDTH. If omitted, it defaults to 0.

y < dimension unit >

Vertical position of the LAYOUT, which will extend down from this

	position for HEIGHT. If omitted, it defaults to the current vertical position on the page.
width	< dimension unit >
	Horizontal width of the LAYOUT. If omitted, it defaults to the maximum horizontal spaced needed to display all regions.
height	< dimension unit >
	Vertical height of the LAYOUT. If omitted, it defaults to the maximum vertical spaced needed to display all regions.
style	< string character variable >
	The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
overrides	< string character variable >
	The style attributes to override those defined in the selected style element.

Example

```
obj.layout_absolute(x: "4in",
                  y: "3in",
                  width: "3in",
                  height: "1in");
obj.region();
obj.format_text(data: "Some text for a region.");
obj.layout_end();
```

REGION

Syntax

Object.region(< optional argument >, ... , < optional argument >);

Description

Optional Arguments

width	Horizontal width of the REGION, and is restricted by the LAYOUT dimensions. If omitted, it defaults to the maximum horizontal spaced needed to display the output contained in the REGION. The sum of all region widths cannot exceed the LAYOUT horizontal dimension.
height	Vertical height of the REGION, and is restricted by the LAYOUT dimensions. If omitted, it defaults to the maximum vertical spaced needed to display the output contained in the REGION. The sum of all region heights cannot exceed the LAYOUT vertical dimension. This option should be used very sparingly.
style	The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
overrides	The style attributes to override those defined in the selected style element.

Example

```
obj.layout_absolute(x: "4in",
                  y: "3in",
                  width: "3in",
```

```
        height: "1in");
    obj.region();
    obj.format_text(data: "Some text for a region.");
obj.layout_end();
```

LAYOUT_END**Syntax**

```
Object.layout_end();
```

Description**No Arguments****Example**

```
obj.layout_end();
```

MISCELLANEOUS METHODS

OPEN_DIR**Syntax**

```
Object.open_dir( < optional argument >, ... , < optional argument > );
```

Description

Open a table of contents directory. The OPEN_DIR is always used in conjunction with the CLOSE_DIR method.

Optional Arguments

name < string | character variable >

The name of the director that will be used in the table of contents (TOC) and DMS Results window.

label The label of the directory that will be used in the table of contents (TOC) and DMS Results window.

by < 1 (True) | 0 (False) >

This is a directory that is related to the unique by value. Special BY-group processing occurs when this option is set.

Example

```
obj.open_dir(name: "Sub Directory");
    obj.format_text(data: "here is some data.");
obj.close_dir();
```

CLOSE_DIR**Syntax**

```
Object.close_dir();
```

Description

Close the open directory. The CLOSE_DIR is always used in conjunction with the OPEN_DIR method.

No Arguments

Example

```
obj.open_dir(name: "Sub Directory");  
  obj.format_text(data: "here is some data.");  
obj.close_dir();
```

LINE

Syntax

Object.line(< optional argument >, ... , < optional argument >);

Description

Draws a horizontal rule (line) across the page.

Optional Arguments

size < dimension unit >

The thickness of the line.

style The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.

Example

```
obj.line();  
obj.line(size: "1mm");
```

IMAGE

Syntax

Object.image(< optional argument >, ... , < optional argument >);

Description

Insert the image into all open output destinations.

Optional Arguments

file < string | character variable >

The FILEREF or physical file name of the external image to include.

style The style element that contains the collection of style attributes to be applied to the data value. The default style element is "text".

overrides < string | character variable >

The style attributes to override those defined in the selected style element.

Example

```
obj.image(file: "c:\someimage.jpg");
```

HREF

Syntax

Object.href(< optional argument >, ... , < optional argument >);

Description

Create a link to another document.

Optional Arguments

data	< string number character variable numeric variable >
	The data value to display. If the data is numeric and no format has been specified the data value will be formatted using the BEST format.
format	< string character variable >
	The SAS format to be applied to the data argument.
href	< string character variable >
	The URL that our data string.
style	< string character variable >
	The style element that contains the collection of style attributes to be applied to the data value. The default style element is TEXT.
split	< string character variable >
	Split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not considered part of the data value.
no_base	< numeric numeric variable >

Example

```
obj.href(data: "SAS Home page",
         file: "http://www.sas.com/index.html");
```

DELETE

Syntax

```
Object.delete( );
```

Description

Delete an instance of the ODSOUT class.

No Arguments