THE POWER TO KNOW.

SAS® 9.2 Output Delivery System

User's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® 9.2 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® 9.2 Output Delivery System: User's Guide

ISBN 978-1-59994-591-0

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, February 2009

1st printing, March 2009

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

What's New

Overview

New and enhanced features in the Output Delivery System (ODS) provide an almost limitless number of choices for reporting and displaying analytical results with a greater variety of formatting selections and output destinations.

- With ODS statements, you can use new ODS packages, measured RTF output, and enhanced inline formatting, among other new features.
- Using the DOCUMENT procedure, you can perform four new tasks.
- Several new features and enhancements have been added to the TEMPLATE procedure, including a new crosstabulation table template and enhanced style inheritance.
- Improved ODS statistical graphics enable you to use ODS and SAS/GRAPH to create and modify statistical graphics.
- SAS/GRAPH uses ODS styles by default for graphical output.
- You can use the PDFSECURITY system option to encrypt and password-protect your PDF files.
- SVG (Scalable Vector Graphics) and new TrueType fonts can be added to ODS output.
- You can use PROC SQL to query open ODS destinations.
- You can control the way you view your PDF document by specifying system options PDFPAGELAYOUT and the PDFPAGEVIEW.

New Features and Enhancements for ODS Statements

The following ODS statements are new:

ODS TAGSETS.RTF
: creates measured RTF output that enables you to specify how and where page breaks occur and when to place titles and footnotes in the body of a page.

ODS PACKAGE
: opens, adds to, publishes, or closes one SAS ODS package object. ODS packages enable ODS destinations to use the SAS Publishing Framework, which is a feature

of SAS Integration Technologies. An ODS package tracks the output from any active destinations that connect with it. After the destinations close, the package can be published to any of the publish destinations. You can use ODS packages with the ODS PACKAGE statement.

ODS TEXT=
inserts text into ODS output.

The following new functionality works with the ODS PRINTER and PDF destinations:

- Style attribute TEXTDECORATION= can now be specified with the PDF destination.
- In addition to supporting HTTP, the URL= style attribute now supports HTTPS, FTP, NEWS, and MAILTO.

New options have been added to the following ODS statements:

- Inline formatting has new syntax for SAS 9.2 and inline styles can now be nested. The ODS ESCAPECHAR statement now supports the list of functions shown in the section "Valid Functions That Can Be Used with ODS ESCAPECHAR" that can be used with various destinations. In addition, you can now use the UNICODE inline formatting function to select any available Unicode character in the current Unicode font.
- The ODS EXCLUDE statement now supports the following options:

  NOWARN
  suppresses the warning that an output object was requested but not created.

  WHERE=
  excludes output objects that meet a particular condition.

- The ODS GRAPHICS statement now supports the following options:

  ANTIALIAS | NOANTIALIAS | ANTIALIAS=
  specifies whether anti-aliasing is applied to the rendering of the line and markers in the graph. Anti-aliasing smooths the appearance of diagonal lines and some markers.

  ANTIALIASMAX=
  specifies the maximum number of markers or lines to be anti-aliased before anti-aliasing is disabled. The default is 600.

  BORDER | NOBORDER | BORDER=
  specifies whether to draw the graph with a border on the outermost layout.

  DISCRETEMAX=
  specifies the maximum number of discrete values to be shown in a plot.

  GROUPMAX=
  specifies the maximum number of group values to be shown in a plot.

  HEIGHT=
  specifies the height of the graph.

  IMAGEFMT=
  specifies the image format to display graphics in ODS output. If the image format is not valid for the active output destination, the device is automatically remapped to the default image format.

  IMAGEMAP | NOIMAGEMAP | IMAGEMAP=
  specifies whether data tips are generated.

IMAGENAME=
specifies the base image filename. By default, the name of the output object is used. You can determine the name of the output object by using the ODS TRACE statement.

LABELMAX=
specifies the maximum number of labeled areas before labeling is disabled.

MAXLEGENDAREA=
specifies an integer that is interpreted as the maximum percentage of the overall graph's area that a legend can occupy.

PANELCELLMAX=
specifies the maximum number of cells in a graph panel where the number of cells is determined dynamically by classification variables.

RESET | RESET=
resets one or more ODS GRAPHICS options to its default.

SCALE | NOSCALE | SCALE=
specifies whether the content of the graph is scaled proportionally.

TIPMAX=
specifies the maximum number of distinct data tip boxes that are allowed before the boxes are disabled.

WIDTH=
specifies the width of the graph.

- The ODS HTML statement now supports the following options:

  DEVICE=
  specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

  OPTIONS=
  specifies tagset-specific suboptions and a named value. The DOC= suboption allows you to produce a list of other suboptions that can be specified for OPTIONS=

- The ODS LISTING statement now supports the following option:

  DEVICE=
  specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

  GPATH=
  specifies the location for all graphics output that is generated while the destination is open.

  PACKAGE
  specifies that the output from the destination be added to a package.

- The ODS MARKUP statement now supports the following options:

  CSSSTYLE=
  specifies a cascading style sheet to apply to your output.

  DEVICE=
  specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

IMAGE_DPI=
specifies the image resolution in dots per inch for output images.

EVENT=
specifies an event and the value for event variables that are associated with the event.

OPTIONS=
specifies tagset-specific suboptions and a named value. The DOC= suboption allows you to produce a list of other suboptions that can be specified for OPTIONS=

PACKAGE
specifies that the output from the destination be added to a package. The PACKAGE option is valid for all markup family statements.

TEXT=
inserts text into your document by triggering the paragraph event and specifying a text string to be assigned to the VALUE event variable.

- The ODS OUTPUT statement now supports the following option:

  NOWARN
  suppresses the warning that an output object was requested but not created.

- The ODS PRINTER, ODS PDF, and ODS PCL statements now support the following options:

  CSSSTYLE=
  specifies a cascading style sheet to apply to your output.

  DPI=
  specifies the image resolution in dots per inch for output images.

  NEWFILE=
  creates a new file at the specified *starting-point*.

  PACKAGE
  specifies that the output from the destination be added to an ODS package.

- The ODS PDF and ODS PRINTER statements now support the following options.

  PDFTOC=
  controls the level of the expansion of the table of contents in PDF documents.

- The ODS RTF statement now supports the following options:

  BODYTITLE
  specifies that SAS titles and footnotes are placed in the body of the RTF document rather than the headers and footers section of the document.

  BODYTITLE_AUX
  specifies that SAS titles and footnotes are placed in the body of the RTF document rather than the headers and footers section of the document. The titles and footnotes are placed in cells, which allows the titles and footnotes to be centered, right-justified and left-justified.

  CONTENTS
  produces a table of contents page that is inserted into the RTF document when the TOC_DATA option is specified.

  CSSSTYLE=
  specifies a cascading style sheet to apply to your output.

DEVICE=
specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

IMAGE_DPI=
specifies the resolution for your graphics.

PACKAGE
specifies the location of an aggregate storage location or a SAS catalog for all RTF files.

TOC_DATA
specifies whether contents data is inserted into the RTF document.

- The ODS SELECT statement now supports the following options:

  NOWARN
  suppresses the warning that an output object was requested but not created.

  WHERE=
  selects output objects that meet a particular condition.

- The ODS TRACE statement now supports the following option:

  EXCLUDED
  includes, in the trace record, information about excluded output objects.

New Features and Enhancements for the DOCUMENT Procedure

The DOCUMENT procedure now enables you to do the following:

- Use all PROC DOCUMENT features with the REPORT procedure.
- Create columns for BY variables in the contents list with the new BYGROUPS option in the LIST statement.
- Conditionally select a subset of entries in an ODS document for copying, listing, deleting, moving, or replaying by using WHERE expressions with these statements:
  - COPY TO
  - DELETE
  - LIST
  - MOVE TO
  - REPLAY
- Specify the level of the path that you want to delete with the new LEVELS= option in the DELETE statement.
- Write, to any open ODS destination, the source code of the ODS template that is associated with the specified output object, with the new OBTEMPL statement.

The DOCUMENT procedure now supports the REPORT procedure.

New Features and Enhancements for the TEMPLATE Procedure

The following general enhancements and features have been added to the TEMPLATE procedure:

- The "LIST Statement" on page 411 now supports WHERE expressions, which enables you to select items for listing that meet a particular condition.
- The "SOURCE Statement" on page 417 now supports WHERE expressions, which enables you to select items that meet a particular condition.
- The TEMPLATE procedure now enables you to customize the appearance of crosstabulation tables that are created with the FREQ procedure. The new CrossTabFreqs table template describes how to display PROC FREQ's crosstabulation table. You can create a customized CrossTabFreqs table template to do the following:
  - use custom formats for cellvalues
  - specify a style for each value in a cell
  - change the stacking order of values in a cell
  - change and style headers and footers
  - use variable labels in headers and footers
  - style table regions independently
  - change or remove the legend

The following enhancements and features have been added to the TEMPLATE procedure for table templates:

- You can now create the master table templates that are globally applied to all of your tabular output:

  Base.Template.Column on page 600

  Base.Template.Footer on page 627

  Base.Template.Header on page 619

  Base.Template.Table on page 642

  These master templates are available in all of the DEFINE statements within a table template.
- You can now use subsetting variables with the "CELLSTYLE AS Statement" on page 614 to find common values in table templates and column templates.
- You can use the TableHeaderContainer and TableFooterContainer style elements along with the border control style attributes to change the borders of the regions that surround the table header and footer.

The following new statements are provided in the TEMPLATE procedure for style definitions:

CLASS
creates a style element from a like-named style element.

IMPORT
imports Cascading Style Sheet (CSS) code from an external file and converts the code to style elements and style attributes that are then included in the style definition.

The following enhancements and features have been added to the TEMPLATE procedure for ODS style definitions:

- You can now create the master template, Base.Template.Style on page 491, that is globally applied to all of your style definitions. Base.Template.Style is created with the "DEFINE STYLE Statement" on page 490.
- Style element inheritance has been enhanced. In addition, the functionality of the REPLACE statement has been replaced by the STYLE statement, and the REPLACE statement is no longer supported. For more information, see "Understanding Inheritance" on page 543.
- You can now use the following style attributes with inline formatting or the TableHeaderContainer and TableFooterContainer style elements to make individual border style changes to RTF output:
  - BORDERBOTTOMCOLOR= on page 511
  - BORDERBOTTOMSTYLE= on page 511
  - BORDERBOTTOMWIDTH= on page 511
  - BORDERCOLOR= on page 511
  - BORDERLEFTSTYLE= on page 512
  - BORDERLEFTCOLOR= on page 512
  - BORDERLEFTWIDTH= on page 512
  - BORDERRIGHTCOLOR= on page 512
  - BORDERRIGHTSTYLE= on page 513
  - BORDERRIGHTWIDTH= on page 513
  - BORDERTOPCOLOR= on page 513
  - BORDERTOPSTYLE= on page 513
  - BORDERTOPWIDTH= on page 514
- The new style HighContrast enables you to produce reports with HTML output in high contrast to meet accessibility requirements.

The STYLE statement now supports the following option:

SELF
: specifies to use the preceding style element or style elements of the same name for the parent of the new style.

The following new statements are provided in the TEMPLATE procedure for the tagset template:

CONTINUE
: specifies that the execution of the DO loop returns to the corresponding DO statement.

DO
: begins a statement block that executes if the required condition is true.

DONE
: ends a statement block.

ELSE
: begins a statement block that executes if the corresponding DO statement is false.

EVAL
: creates or updates a user-defined variable and its value.

ITERATE
specifies a dictionary variable or list variable to loop through, and for each iteration, assigns the variable's values to the _NAME_ and _VALUE_ event variables.

NEXT
increments a dictionary or list variable to the next value.

STOP
moves execution to the end of the current statement block.

The following enhancements has been added to the TEMPLATE procedure for tagset templates:

- Stream commands in tagset templates are able to specify variables.
- You can now use the master template Base.Template.Tagset that is globally applied to all of your tagsets. Base.Template.Tagset is created with the DEFINE TAGSET statement.

Improved ODS Statistical Graphics

The ODS graphics capability, which was experimental for SAS 9.1, is now production for SAS 9.2 as ODS graphics.

- There are over 50 procedures in SAS/STAT, SAS/ETS, SAS/QC, and Base SAS that have been modified to use ODS graphics. Many new plots are now produced by these procedures, either by default or with the specification of procedure options.
- The functionality of ODS graphics has been extended with the addition of new graph types, ODS styles designed for statistical work, and a point-and-click editor for enhancing titles, labels, and other graph features.
- You can also modify graphs by changing their underlying templates, which are supplied by SAS and are written with PROC TEMPLATE and the Graph Template Language (GTL).
- The LISTING destination is now supported by ODS graphics.
- A new family of SAS/GRAPH procedures uses ODS graphics to create stand-alone plots, such as scatterplots overlaid with smoothers, which are particularly useful for exploratory data analysis.
- The new SGRENDER procedure provides a way to create customized displays by writing your own templates with the GTL and PROC TEMPLATE.

For complete documentation about ODS graphics, see the *SAS/GRAPH: Statistical Graphics Procedures Guide*, *SAS/GRAPH: Graph Template Language Reference*, and the *SAS/GRAPH: Graph Template Language User's Guide*.

Note: A SAS/GRAPH license is now required to use ODS graphics. △

New ODS Support for SAS/GRAPH

All SAS/GRAPH procedures and devices now support ODS styles. By default, all colors, fonts, symbols, and graph sizes are derived from the current style. Procedure statement options and SAS/GRAPH GOPTIONS can still be used to override individual elements of the graph, providing you the flexibility to customize the appearance of any graph.

Additionally, the colors used by the styles have been updated to enhance the appearance of your graphics output.

If you have multiple ODS destinations open, SAS/GRAPH automatically selects the appropriate device for each destination. In addition, each graph uses the ODS style that is associated with each destination. You do not need to specify a device or style to get optimal results. For example, if you do not specify a device, then SAS/GRAPH automatically selects the PNG device for the HTML destination and the SASEMF device for the RTF destination.

Also, if you have multiple ODS destinations open and you are using a device other than the Java or ActiveX devices (ACTIVEX, JAVA, ACTXIMG, or JAVAIMG), a different GRSEG is created for each open destination. The GRSEG for the first destination is stored in WORK.GSEG. The GRSEGs for any other open destinations are named according to the destination (for example, WORK.HTML).

New PDF Security

You can now encrypt and password-protect your PDF files easily in SAS with the PDFSECURITY system option. See the PDFSECURITY system option in *SAS Language Reference: Dictionary*.

New Scalable Vector Graphics and Fonts

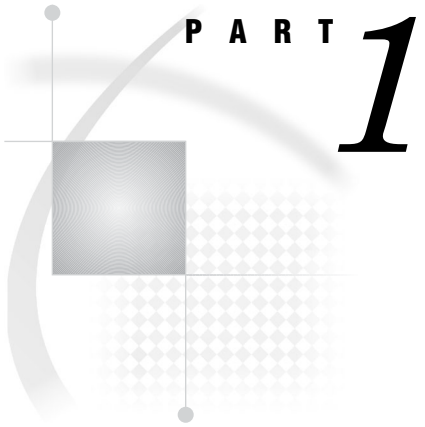
ODS styles can now use new TrueType fonts. All Universal Printers and many SAS/GRAPH devices use the FreeType library to render TrueType fonts for output in all of the operating environments that SAS software supports. In addition, by default, many SAS/GRAPH device drivers and all Universal Printers generate output using ODS styles, and these ODS styles use TrueType fonts. In addition to SAS Monospace and SAS Monospace Bold, 21 new TrueType fonts are made available when you install SAS:

- five Latin fonts compatible with Microsoft
- eight multilingual Unicode fonts
- eight monolingual Asian fonts

ODS now supports Scalable Vector Graphics, which is a language for describing two-dimensional graphics and graphical applications in XML. For more information about Scalable Vector Graphics and the TrueType fonts, see the section "Printing with SAS" in *SAS Language Reference: Concepts*.

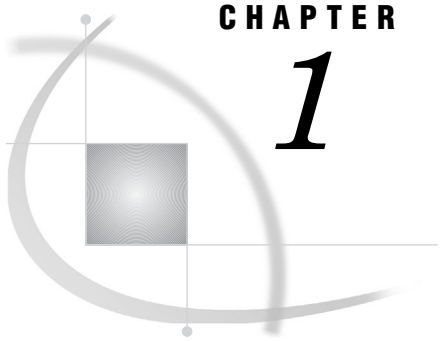
Query Open ODS Destinations

You can now programmatically query SAS for open ODS destinations with the new SQL dictionary table DESTINATIONS and its associated view. See the section "Using the DICTIONARY Tables" in the SQL Procedure in the *Base SAS Procedures Guide*.



PART 1

Introduction



CHAPTER

1

Getting Started with the Output Delivery System

Welcome to the Output Delivery System

Before SAS 7, most SAS procedures generated output that was designed for a traditional line-printer. This type of output has limitations that prevent you from getting the most value from your results:

- Traditional SAS output is limited to monospace fonts. In a time of desktop document editors and publishing systems, you want more versatility in printed output.
- Some commonly used procedures produce printed output but do not create an output data set. Many times it would be very convenient to produce not only printed output but also an output data set that you could use as input to another SAS procedure or to a DATA step.

ODS is designed to overcome these limitations and make it easier for you to format your output. The SAS Output Delivery System (ODS) gives you greater flexibility in generating, storing, and reproducing SAS procedure and DATA step output along with a wide range of formatting options. ODS provides formatting functionality that is not available when using individual procedures or the DATA step without ODS.

Accessibility Features in ODS

The Output Delivery System conforms to the U.S. Section 508 guidelines for Web-based content. If you have specific questions about the accessibility of SAS products, send them to accessibility@sas.com or call SAS Technical Support.

The following additional accessibility items are available as programming options:

"Event Variables" on page 833

ABBR
specifies an abbreviation for the event variable.

ACRONYM
specifies an acronym for an event variable.

ALT
specifies an alternate description of an event variable.

CAPTION
specifies the caption for a table.

LONGDESC
specifies the long description of an event variable.

SUMMARY
specifies a summary of a table.

Style Template:

STYLES.HIGHCONTRAST
creates the same output as the default output except all of the colors are black on white.

"Header Attributes" on page 628

ABBR= on page 630
specifies an abbreviation for the header.

ACRONYM= on page 630
specifies an acronym for the header.

ALT= on page 630
specifies an alternate description of the header.

GENERIC on page 632
specifies whether multiple columns can use the header.

LONGDESC= on page 632
specifies the long description of the header.

"Table Attributes" on page 642

LONGDESC= on page 648
specifies a long description of a table.

ALT= on page 645
specifies an alternate description of a table.

The following tagsets and ODS statements are 508 compliant:

"ODS PHTML Statement" on page 215
opens, manages, or closes the PHTML destination, which produces simple HTML output that uses twelve style elements and no class attributes.

"ODS HTMLCSS Statement" on page 135
opens, manages, or closes the HTMLCSS destination, which produces HTML output with cascading style sheets.

"ODS HTML Statement" on page 124
opens, manages, or closes the HTML destination, which produces HTML 4.0 output that contains embedded style sheets.

MSOFFICE2K on page 282 tagset
produces HTML code for output generated by ODS for Microsoft Office products.

In SAS 9.1 and later releases, all of the accessibility enhancements have been merged into the ODS HTML tagsets. No additional steps are required.

A Quick Start to Using ODS

The Purpose of These Examples

The following examples are designed to help you to begin using ODS quickly. Use them to learn how to produce output that contains more interesting formatting. Then, to learn more about the depth, breadth, and true power of ODS, see "Introduction to the Output Delivery System" on page 16.

Creating Listing Output

Creating the listing output is simple—just run a DATA step or PROC step as usual. By default, the LISTING destination is on, and the DATA step and Base SAS procedures create listing output through ODS:

```
options source pagesize=60 linesize=80 nodate;

data employee_data;
   input IDNumber $ 1-4 LastName $ 9-19 FirstName $ 20-29
         City $ 30-42 State $ 43-44 /
         Gender $ 1 JobCode $ 9-11 Salary 20-29 @30 Birth date9.
         @43 Hired date9. HomePhone $ 54-65;
   format birth hired date9.;

            datalines;
1919    Adams      Gerald    Stamford     CT
M       TA2        34376     15SEP48      07JUN75    203/781-1255
1653    Alexander  Susan     Bridgeport   CT
F       ME2        35108     18OCT52      12AUG78    203/675-7715
1400    Apple      Troy      New York     NY
M       ME1        29769     08NOV55      19OCT78    212/586-0808
1350    Arthur     Barbara   New York     NY
F       FA3        32886     03SEP53      01AUG78    718/383-1549
1401    Avery      Jerry     Paterson     NJ
M       TA3        38822     16DEC38      20NOV73    201/732-8787
1499    Barefoot   Joseph    Princeton    NJ
M       ME3        43025     29APR42      10JUN68    201/812-5665
1101    Baucom     Walter    New York     NY
M       SCP        18723     09JUN50      04OCT78    212/586-8060
1333    Blair      Justin    Stamford     CT
M       PT2        88606     02APR49      13FEB69    203/781-1777
1402    Blalock    Ralph     New York     NY
M       TA2        32615     20JAN51      05DEC78    718/384-2849
1479    Bostic     Marie     New York     NY
F       TA3        38785     25DEC56      08OCT77    718/384-8816
1403    Bowden     Earl      Bridgeport   CT
M       ME1        28072     31JAN57      24DEC79    203/675-3434
```

```
1739    Boyce      Jonathan  New York     NY
M       PT1        66517     28DEC52      30JAN79    212/587-1247
1658    Bradley    Jeremy    New York     NY
M       SCP        17943     11APR55      03MAR80    212/587-3622
1428    Brady      Christine Stamford     CT
F       PT1        68767     07APR58      19NOV79    203/781-1212
1407    Grant      Daniel    Mt. Vernon   NY
M       PT1        68096     26MAR57      21MAR78    914/468-1616
1114    Green      Janice    New York     NY
F       TA2        32928     21SEP57      30JUN75    212/588-1092
;

proc print data=employee_data(obs=12);
   id idnumber;
   title 'Personnel Data';
run;
```

Output 1.1 Listing Output

```
                          Personnel Data                                1

  ID                 First                                        Job
Number    LastName   Name          City       State    Gender    Code

  1919    Adams      Gerald      Stamford       CT        M       TA2
  1653    Alexander  Susan       Bridgeport     CT        F       ME2
  1400    Apple      Troy        New York       NY        M       ME1
  1350    Arthur     Barbara     New York       NY        F       FA3
  1401    Avery      Jerry       Paterson       NJ        M       TA3
  1499    Barefoot   Joseph      Princeton      NJ        M       ME3
  1101    Baucom     Walter      New York       NY        M       SCP
  1333    Blair      Justin      Stamford       CT        M       PT2
  1402    Blalock    Ralph       New York       NY        M       TA2
  1479    Bostic     Marie       New York       NY        F       TA3
  1403    Bowden     Earl        Bridgeport     CT        M       ME1
  1739    Boyce      Jonathan    New York       NY        M       PT1

  ID
Number    Salary       Birth        Hired     HomePhone

  1919      34376    15SEP1948    07JUN1975    203/781-1255
  1653      35108    18OCT1952    12AUG1978    203/675-7715
  1400      29769    08NOV1955    19OCT1978    212/586-0808
  1350      32886    03SEP1953    01AUG1978    718/383-1549
  1401      38822    16DEC1938    20NOV1973    201/732-8787
  1499      43025    29APR1942    10JUN1968    201/812-5665
  1101      18723    09JUN1950    04OCT1978    212/586-8060
  1333      88606    02APR1949    13FEB1969    203/781-1777
  1402      32615    20JAN1951    05DEC1978    718/384-2849
  1479      38785    25DEC1956    08OCT1977    718/384-8816
  1403      28072    31JAN1957    24DEC1979    203/675-3434
  1739      66517    28DEC1952    30JAN1979    212/587-1247
```

Listing output is the default format; therefore, when you request another format, your programs will create both listing output and output in the requested format. To prevent listing output from being created, use this statement:

```
ods listing close;
```

Creating Output in HTML Format

If you want to display output from a SAS program from the Web, you can use ODS to create output that is formatted in Hypertext Markup Language (HTML). To create HTML output, use the ODS HTML statement:

```
ods html file='external-file-for-HTML-output';
```

If you do not want to generate listing output in addition to the HTML output, then use this statement:

```
ods listing close;
```

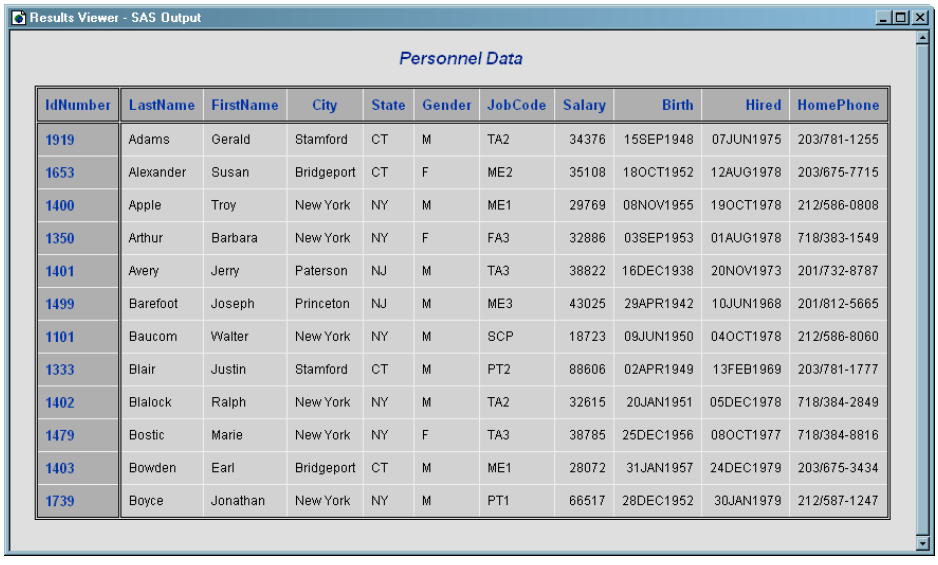
The following program contains a PROC PRINT step that produces output in HTML, but does not produce the default listing output. You can browse this output with Internet Explorer, Netscape, or any other browser that fully supports HTML 3.2 or later.

```
ods listing close;
ods html file='external-file-for-HTML-output';


proc print data=employee_data(obs=12);
   id idnumber;
   title 'Personnel Data';
run;

ods html close;
ods listing;
```

Note the two ODS statements that follow the PROC PRINT step. To be able to browse your HTML files in a browser, you must execute the ODS HTML CLOSE statement. It is simply good practice to reset ODS to listing output, which is the default setting.

Display 1.1 HTML 3.2 Output

The following output is formatted in HTML 3.2 output and viewed in an Internet Explorer 5.0 browser.



IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	15SEP1948	07JUN1975	203/781-1255
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108	18OCT1952	12AUG1978	203/675-7715
1400	Apple	Troy	New York	NY	M	ME1	29769	08NOV1955	19OCT1978	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	03SEP1953	01AUG1978	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	16DEC1938	20NOV1973	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	29APR1942	10JUN1968	201/812-5665
1101	Baucom	Walter	New York	NY	M	SCP	18723	09JUN1950	04OCT1978	212/586-8060
1333	Blair	Justin	Stamford	CT	M	PT2	88606	02APR1949	13FEB1969	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	20JAN1951	05DEC1978	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	25DEC1956	08OCT1977	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	31JAN1957	24DEC1979	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	28DEC1952	30JAN1979	212/587-1247

Producing Output in Multiple Formats at the Same Time

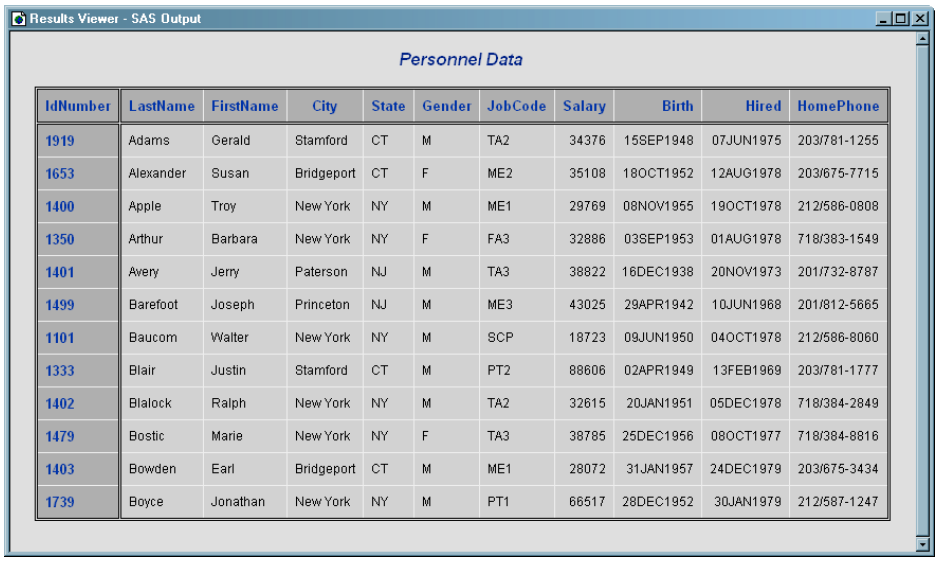
A simple way to produce output in multiple formats at one time is to produce the default listing output and then request an additional format, such as HTML, PDF, RTF, or PostScript.

```
ods html file='HTML-file-pathname.html';
ods pdf file='PDF-file-pathname.pdf';
ods rtf file='RTF-file-pathname.rtf';
ods ps file='PS-file-pathname.ps';

proc print data=employee_data(obs=12);
   id idnumber;
   title 'Personnel Data';
run;

ods  _all_  close;
ods listing;
```

Note the two ODS statements that follow the PROC statement. The first one closes all files so that you can use them (for example, you could browse the HTML file or send the PDF file to a printer). The final statement opens the LISTING destination so that ODS returns to producing listing output for subsequent DATA or PROC steps in the current session.

Display 1.2 HTML 3.2 Output

The following output is formatted in HTML 3.2 output and viewed in an Internet Explorer 5.0 browser.

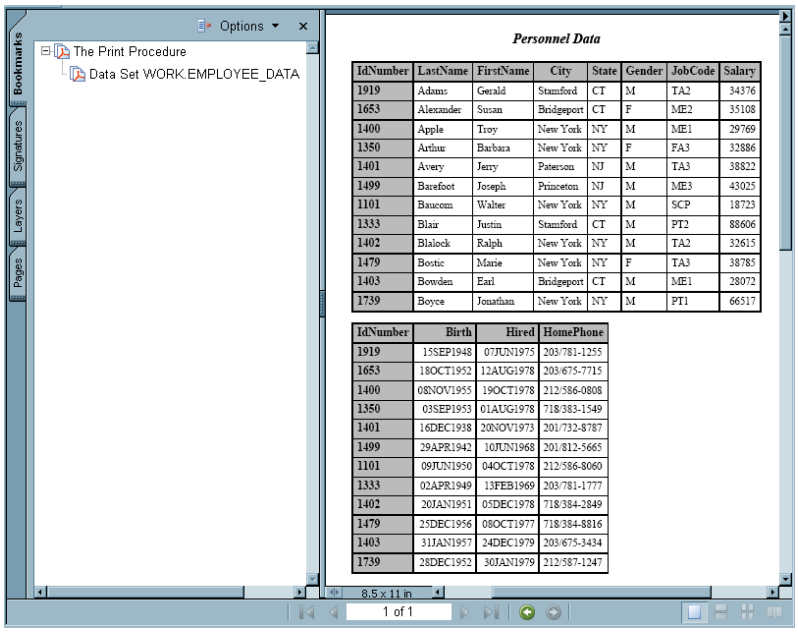


Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	15SEP1948	07JUN1975	203/781-1255
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108	18OCT1952	12AUG1978	203/675-7715
1400	Apple	Troy	New York	NY	M	ME1	29769	08NOV1955	19OCT1978	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	03SEP1953	01AUG1978	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	16DEC1938	20NOV1973	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	29APR1942	10JUN1968	201/812-5665
1101	Baucom	Walter	New York	NY	M	SCP	18723	09JUN1950	04OCT1978	212/586-8060
1333	Blair	Justin	Stamford	CT	M	PT2	88606	02APR1949	13FEB1969	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	20JAN1951	05DEC1978	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	25DEC1956	08OCT1977	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	31JAN1957	24DEC1979	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	28DEC1952	30JAN1979	212/587-1247

Display 1.3 PDF Output

The following output is formatted in PDF and viewed with Adobe Acrobat Reader.



Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary
1919	Adams	Gerald	Stamford	CT	M	TA2	34376
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108
1400	Apple	Troy	New York	NY	M	ME1	29769
1350	Arthur	Barbara	New York	NY	F	FA3	32886
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025
1101	Baucom	Walter	New York	NY	M	SCP	18723
1333	Blair	Justin	Stamford	CT	M	PT2	88606
1402	Blalock	Ralph	New York	NY	M	TA2	32615
1479	Bostic	Marie	New York	NY	F	TA3	38785
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072
1739	Boyce	Jonathan	New York	NY	M	PT1	66517

IdNumber	Birth	Hired	HomePhone
1919	15SEP1948	07JUN1975	203/781-1255
1653	18OCT1952	12AUG1978	203/675-7715
1400	08NOV1955	19OCT1978	212/586-0808
1350	03SEP1953	01AUG1978	718/383-1549
1401	16DEC1938	20NOV1973	201/732-8787
1499	29APR1942	10JUN1968	201/812-5665
1101	09JUN1950	04OCT1978	212/586-8060
1333	02APR1949	13FEB1969	203/781-1777
1402	20JAN1951	05DEC1978	718/384-2849
1479	25DEC1956	08OCT1977	718/384-8816
1403	31JAN1957	24DEC1979	203/675-3434
1739	28DEC1952	30JAN1979	212/587-1247

Display 1.4 RTF Output

The following RTF output is viewed with Microsoft Word 2000.

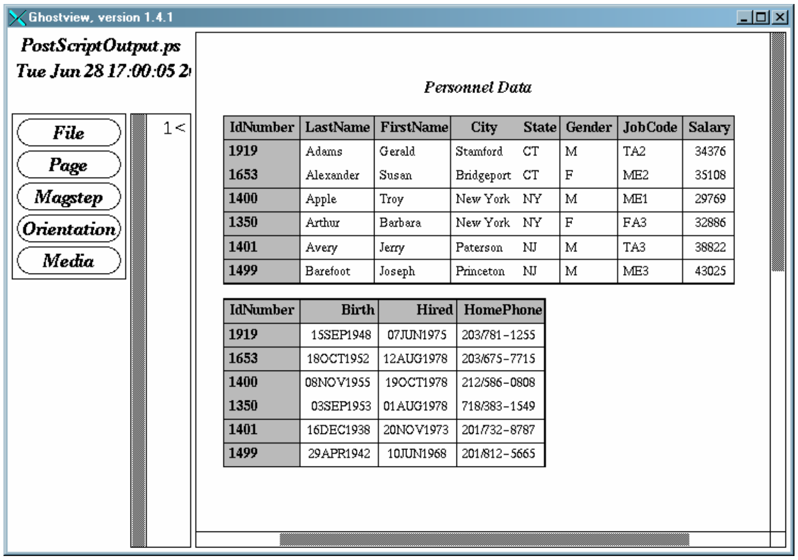
1

Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	15SEP1948	07JUN1975	203/781-1255
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108	18OCT1952	12AUG1978	203/675-7715
1400	Apple	Troy	New York	NY	M	ME1	29769	08NOV1955	19OCT1978	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	03SEP1953	01AUG1978	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	16DEC1938	20NOV1973	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	29APR1942	10JUN1968	201/812-5665
1101	Baucom	Walter	New York	NY	M	SCP	18723	09JUN1950	04OCT1978	212/586-8060
1333	Blair	Justin	Stamford	CT	M	PT2	88606	02APR1949	13FEB1969	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	20JAN1951	05DEC1978	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	25DEC1956	08OCT1977	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	31JAN1957	24DEC1979	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	28DEC1952	30JAN1979	212/587-1247

Display 1.5 PostScript Output

The following PostScript output is viewed with Ghostview.



IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary
1919	Adams	Gerald	Stamford	CT	M	TA2	34376
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108
1400	Apple	Troy	New York	NY	M	ME1	29769
1350	Arthur	Barbara	New York	NY	F	FA3	32886
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025

IdNumber	Birth	Hired	HomePhone
1919	15SEP1948	07JUN1975	203/781-1255
1653	18OCT1952	12AUG1978	203/675-7715
1400	08NOV1955	19OCT1978	212/586-0808
1350	03SEP1953	01AUG1978	718/383-1549
1401	16DEC1938	20NOV1973	201/732-8787
1499	29APR1942	10JUN1968	201/812-5665

Output 1.2 Listing Output

This output is traditional SAS listing output.

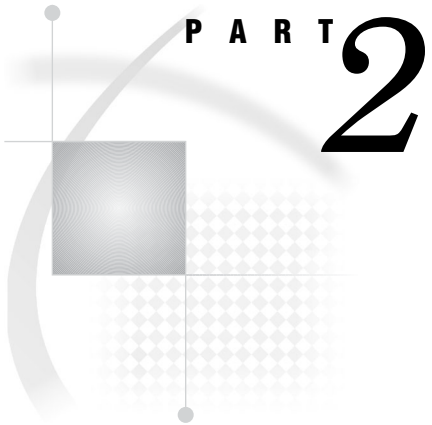
```
                                        Personnel Data                                        5

   ID          First                            Job
 Number LastName Name         City    State Gender Code Salary     Birth     Hired  HomePhone

  1919  Adams     Gerald   Stamford    CT     M    TA2   34376 15SEP1948 07JUN1975 203/781-1255
  1653  Alexander Susan    Bridgeport  CT     F    ME2   35108 18OCT1952 12AUG1978 203/675-7715
  1400  Apple     Troy     New York    NY     M    ME1   29769 08NOV1955 19OCT1978 212/586-0808
  1350  Arthur    Barbara  New York    NY     F    FA3   32886 03SEP1953 01AUG1978 718/383-1549
  1401  Avery     Jerry    Paterson    NJ     M    TA3   38822 16DEC1938 20NOV1973 201/732-8787
  1499  Barefoot  Joseph   Princeton   NJ     M    ME3   43025 29APR1942 10JUN1968 201/812-5665
  1101  Baucom    Walter   New York    NY     M    SCP   18723 09JUN1950 04OCT1978 212/586-8060
  1333  Blair     Justin   Stamford    CT     M    PT2   88606 02APR1949 13FEB1969 203/781-1777
  1402  Blalock   Ralph    New York    NY     M    TA2   32615 20JAN1951 05DEC1978 718/384-2849
  1479  Bostic    Marie    New York    NY     F    TA3   38785 25DEC1956 08OCT1977 718/384-8816
  1403  Bowden    Earl     Bridgeport  CT     M    ME1   28072 31JAN1957 24DEC1979 203/675-3434
  1739  Boyce     Jonathan New York    NY     M    PT1   66517 28DEC1952 30JAN1979 212/587-1247
```

Where to Go from Here

- *Examples of ODS output:* To see the types of output that you can create with ODS, see “Gallery of ODS Samples” on page 16.
- *Essential concepts in ODS:* For concepts that will help you to understand and to use ODS to your best advantage, see “Introduction to the Output Delivery System” on page 16.
- *Creating more complex HTML pages:* With ODS, you can create HTML pages that include a frame and a table of contents. For more information, see “ODS HTML

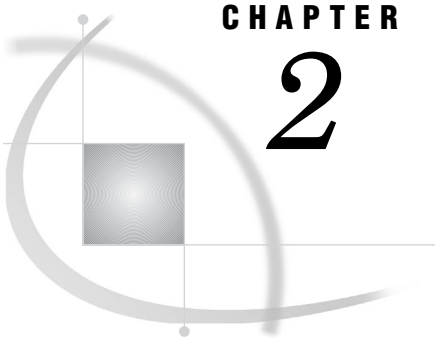
Statement" on page 124 and Appendix 2, "ODS and the HTML Destination," on page 891. You can see many examples of HTML output in the *Base SAS Procedures Guide* online documentation.

- *ODS statements:* For reference information on the ODS statements, see Chapter 5, "Dictionary of ODS Language Statements," on page 67. These statements control the many features of the Output Delivery System.
- *Using ODS with the DATA step:* With the addition of ODS-related options to the FILE and PUT statements, you can use ODS to produce enhanced DATA step reports. See Chapter 3, "Output Delivery System and the DATA Step," on page 39.
- *Creating your own templates:* For even more control over formatting, you can create your own templates for formatting output. See Chapter 7, "TEMPLATE Procedure: Overview," on page 395.



PART 2

Concepts



CHAPTER

2

Output Delivery System: Basic Concepts

Introduction to the Output Delivery System

The Output Delivery System (ODS) gives you greater flexibility in generating, storing, and reproducing SAS procedures and DATA step output, with a wide range of formatting options. ODS provides formatting functionality that is not available from individual procedures or from the DATA step alone. ODS overcomes these limitations and enables you to format your output more easily.

Before SAS 7, most SAS procedures generated output that was designed for a traditional line-printer. This type of output has limitations that prevents you from getting the most value from your results:

- Traditional SAS output is limited to monospace fonts. With today's desktop document editors and publishing systems, you need more versatility in printed output.
- Some commonly used procedures do not produce output data sets. Before ODS, if you wanted to use output from one of these procedures as input to another procedure, then you relied on PROC PRINTTO and the DATA step to retrieve results.

Gallery of ODS Samples

Introduction to the ODS Samples

This section shows you samples of the different kinds of formatted output that you can produce with ODS. The input file contains sales records for TruBlend Coffee Makers, a company that distributes coffee machines.

Listing Output

Traditional SAS output is listing output. You do not need to change your SAS programs to create listing output. By default, you continue to create this kind of output even if you also create a type of output that contains more formatting.

Output 2.1 Listing Output

Average Quarterly Sales Amount by Each Sales Representative 1

---------- Quarter=1 ----------

The MEANS Procedure

Analysis Variable : AmountSold

SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	8	8	14752.5	22806.1	495.0	63333.7
Hollingsworth	5	5	11926.9	12165.2	774.3	31899.1
Jensen	5	5	10015.7	8009.5	3406.7	20904.8

Average Quarterly Sales Amount by Each Sales Representative 2

---------- Quarter=2 ----------

The MEANS Procedure

Analysis Variable : AmountSold

SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	6	6	18143.3	20439.6	1238.8	53113.6
Hollingsworth	6	6	16026.8	14355.0	1237.5	34686.4
Jensen	6	6	12455.1	12713.7	1393.7	34376.7

Average Quarterly Sales Amount by Each Sales Representative 3

---------- Quarter=3 ----------

The MEANS Procedure

Analysis Variable : AmountSold

SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	21	21	10729.8	11457.0	2787.3	38712.5
Hollingsworth	15	15	7313.6	7280.4	1485.0	30970.0
Jensen	21	21	10585.3	7361.7	2227.5	27129.7

Average Quarterly Sales Amount by Each Sales Representative 4

---------- Quarter=4 ----------

The MEANS Procedure

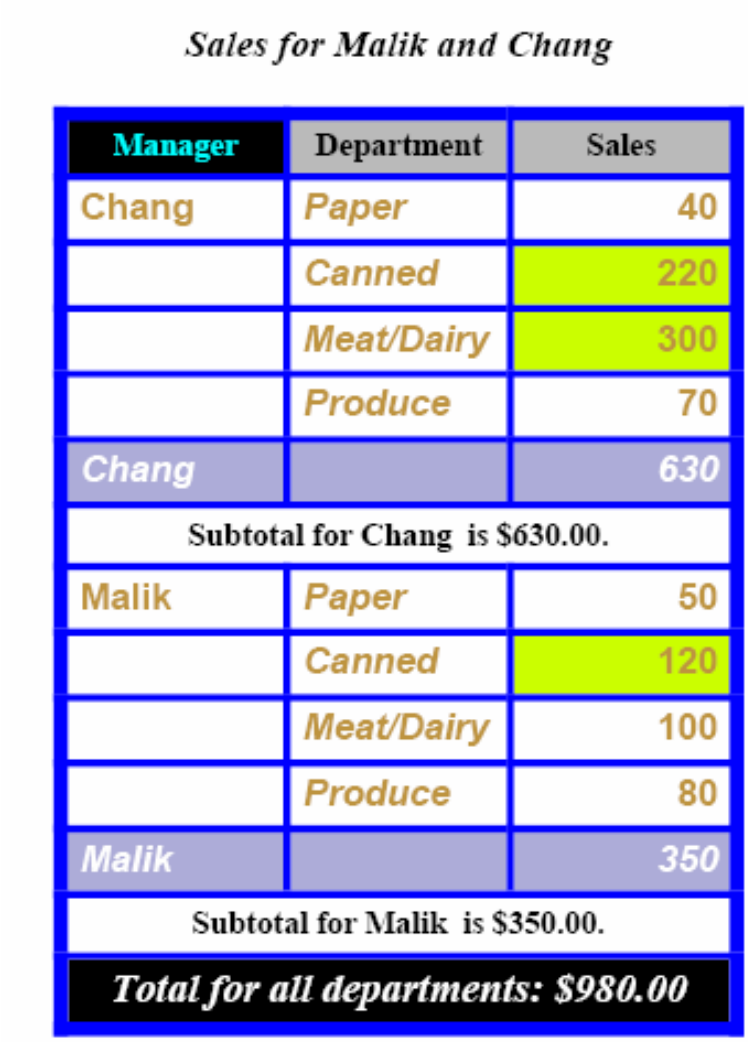
Analysis Variable : AmountSold

SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	5	5	11973.0	10971.8	3716.4	30970.0
Hollingsworth	6	6	13624.4	12624.6	5419.8	38093.1
Jensen	6	6	19010.4	15441.0	1703.4	38836.4

PostScript Output

With ODS, you can produce output in PostScript format.

Display 2.1 PostScript Output



Sales for Malik and Chang

Manager	Department	Sales
Chang	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Chang		630
Subtotal for Chang is $630.00.		
Malik	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Malik		350
Subtotal for Malik is $350.00.		
Total for all departments: $980.00		

HTML Output

With ODS, you can produce output in HTML (Hypertext Markup Language.) You can browse these files with Internet Explorer, Netscape, or any other browser that fully supports HTML 4.0.

Note: To create HTML 3.2 output, use the ODS HTML3 statement. △

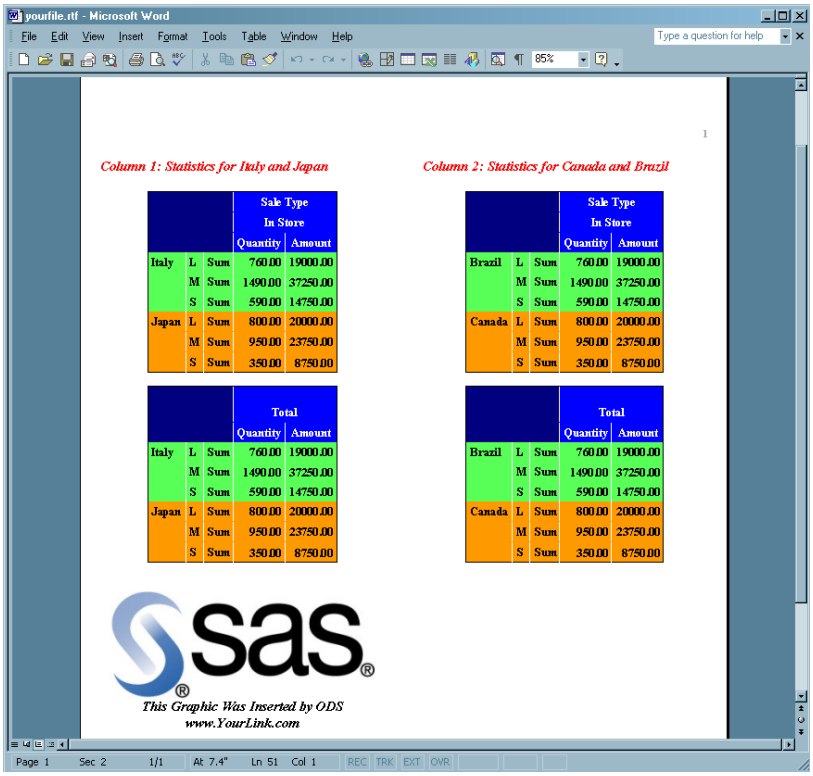
Display 2.2 HTML Output Viewed with Microsoft Internet Explorer

Region by Citysize by Saletype		Saletype				Total	
		Retail		Wholesale			
		Quantity	Amount	Quantity	Amount	Quantity	Amount
		Sum	Sum	Sum	Sum	Sum	Sum
Region	Citysize						
Brazil	L	Missing	Missing	2,272	$45,440	2,272	$45,440
	M	1,066	$26,600	1,066	$21,320	2,132	$47,920
	S	472	$11,800	472	$9,440	944	$21,240
	Total	1,538	$38,400	3,810	$76,200	5,348	$114,600
Canada	Citysize						
	L	2,421	$60,525	2,421	$48,420	4,842	$108,945
	M	1,825	$45,625	1,825	$36,500	3,650	$82,125
	S	623	$15,575	623	$12,460	1,246	$28,035
	Total	4,869	$121,725	4,869	$97,380	9,738	$219,105
France	Citysize						
	L	2,303	$57,575	2,303	$46,060	4,606	$103,635
	M	2,149	$54,725	2,149	$42,980	4,298	$97,705
	S	1,254	$31,150	Missing	Missing	1,254	$31,150
	Total	5,706	$143,450	4,452	$89,040	10,158	$232,490
Mexico	Citysize						
	L	2,655	$66,375	2,655	$53,100	5,310	$119,475
	M	2,360	$59,000	2,360	$47,200	4,720	$106,200
	S	561	$14,025	561	$11,220	1,122	$25,245
	Total	5,576	$139,400	5,576	$111,520	11,152	$250,920
Total	Citysize						
	L	7,379	$184,475	9,651	$193,020	17,030	$377,495
	M	7,400	$185,950	7,400	$148,000	14,800	$333,950
	S	2,910	$72,550	1,656	$33,120	4,566	$105,670
	Total	17,689	$442,975	18,707	$374,140	36,396	$817,115

RTF Output

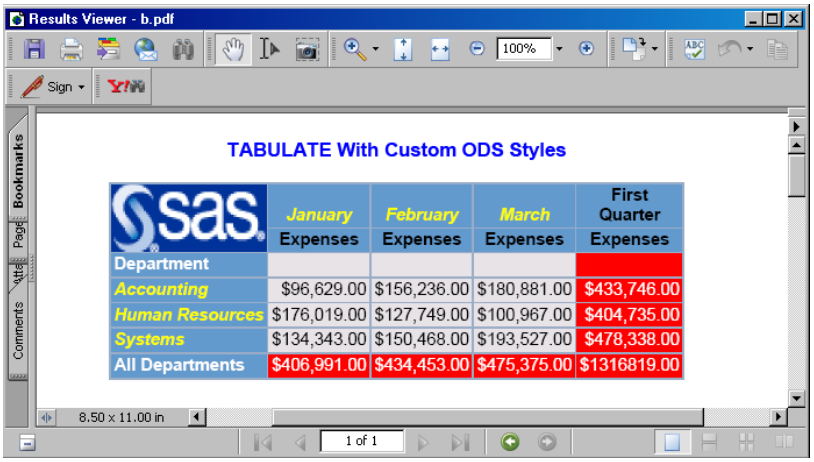
With ODS, you can produce RTF (Rich Text Format) output which is used with Microsoft Word.

Display 2.3 RTF Output Viewed with Microsoft Word



PDF Output

With ODS, you can produce output in PDF (Portable Document Format), which can be viewed with Adobe Acrobat.



XML Output

With ODS, you can produce output that is tagged with Extensible Markup Language (XML) tags.

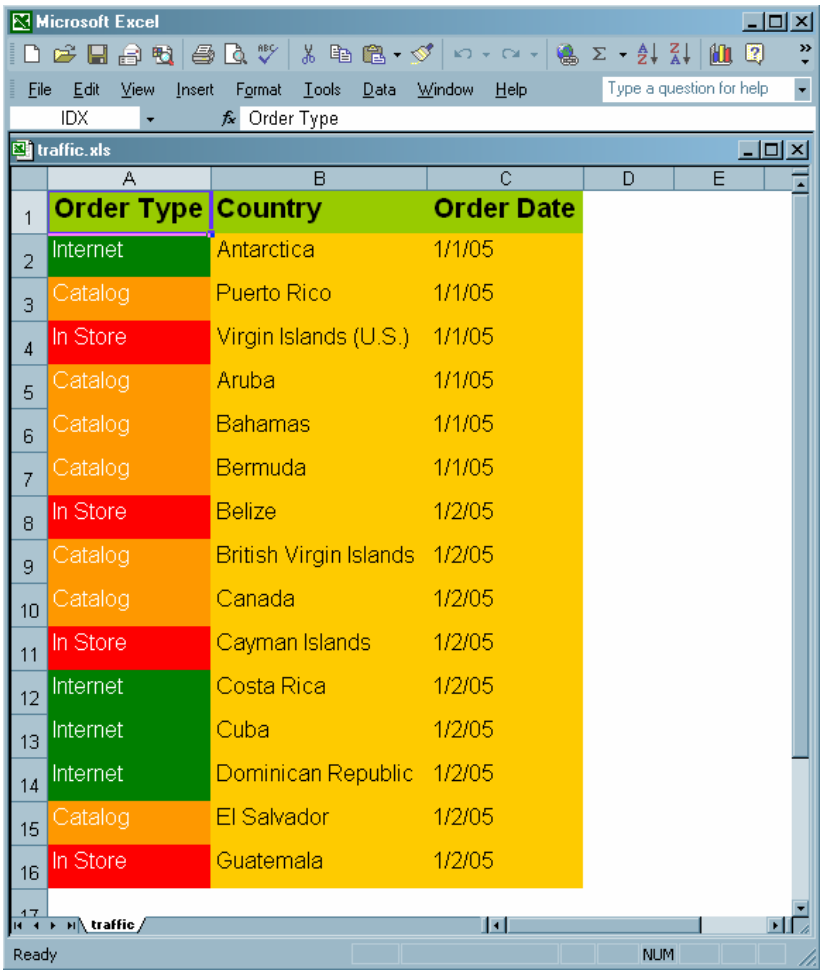
Output 2.2 XML Output file

```
<?xml version="1.0" encoding="windows-1252"?>

<odsxml>
<head>
<meta operator="user"/>
</head>
<body>
<proc name="Print">
<label name="IDX"/>
<title class="SystemTitle" toc-level="1">US Census of Population and Housing</title>
<branch name="Print" label="The Print Procedure" class="ContentProcName" toc-level="1">
<leaf name="Print" label="Data Set SASHELP.CLASS" class="ContentItem" toc-level="2">
<output name="Print" label="Data Set SASHELP.CLASS" clabel="Data Set SASHELP.CLASS">
<output-object type="table" class="Table">

  <style>
    <border spacing="1" padding="7" rules="groups" frame="box"/>
  </style>
<colspecs columns="6">
<colgroup>
<colspec name="1" width="2" align="right" type="int"/>
</colgroup>
<colgroup>
<colspec name="2" width="7" type="string"/>
<colspec name="3" width="1" type="string"/>
<colspec name="4" width="2" align="decimal" type="double"/>
<colspec name="5" width="4" align="decimal" type="double"/>
<colspec name="6" width="5" align="decimal" type="double"/>
</colgroup>
</colspecs>
<output-head>
<row>
<header type="string" class="Header" row="1" column="1">
<value>Obs</value>
</header>
<header type="string" class="Header" row="1" column="2">
<value>Name</value>
</header>
<header type="string" class="Header" row="1" column="3">
<value>Sex</value>
</header>
<header type="string" class="Header" row="1" column="4">
<value>Age</value>
</header>
<header type="string" class="Header" row="1" column="5">
<value>Height</value>
</header>
<header type="string" class="Header" row="1" column="6">
<value>Weight</value>
</header>
</row>
</output-head>
<output-body>
<row>
<header type="double" class="RowHeader" row="2" column="1">
<value> 1</value>
</header>
<data type="string" class="Data" row="2" column="2">
<value>Alfred</value>
</data>
... more xml tagged output...
<
/odsxml>
```

Excel Output

With ODS, you can produce tabular output , which can be viewed with Excel.

Display 2.4 Markup Destination Output Viewed with Excel



	A	B	C	D	E
1	Order Type	Country	Order Date		
2	Internet	Antarctica	1/1/05		
3	Catalog	Puerto Rico	1/1/05		
4	In Store	Virgin Islands (U.S.)	1/1/05		
5	Catalog	Aruba	1/1/05		
6	Catalog	Bahamas	1/1/05		
7	Catalog	Bermuda	1/1/05		
8	In Store	Belize	1/2/05		
9	Catalog	British Virgin Islands	1/2/05		
10	Catalog	Canada	1/2/05		
11	In Store	Cayman Islands	1/2/05		
12	Internet	Costa Rica	1/2/05		
13	Internet	Cuba	1/2/05		
14	Internet	Dominican Republic	1/2/05		
15	Catalog	El Salvador	1/2/05		
16	In Store	Guatemala	1/2/05		

Overview of How ODS Works

Components of SAS Output

The PROC or DATA step supplies raw data and the name of the table template that contains the formatting instructions. ODS formats the output. You can use ODS to format output from individual procedures and from the DATA step in many different forms other than the default SAS listing output.

The following figure shows how SAS produces ODS output.

Figure 2.1 ODS Processing: What Goes in and What Comes Out

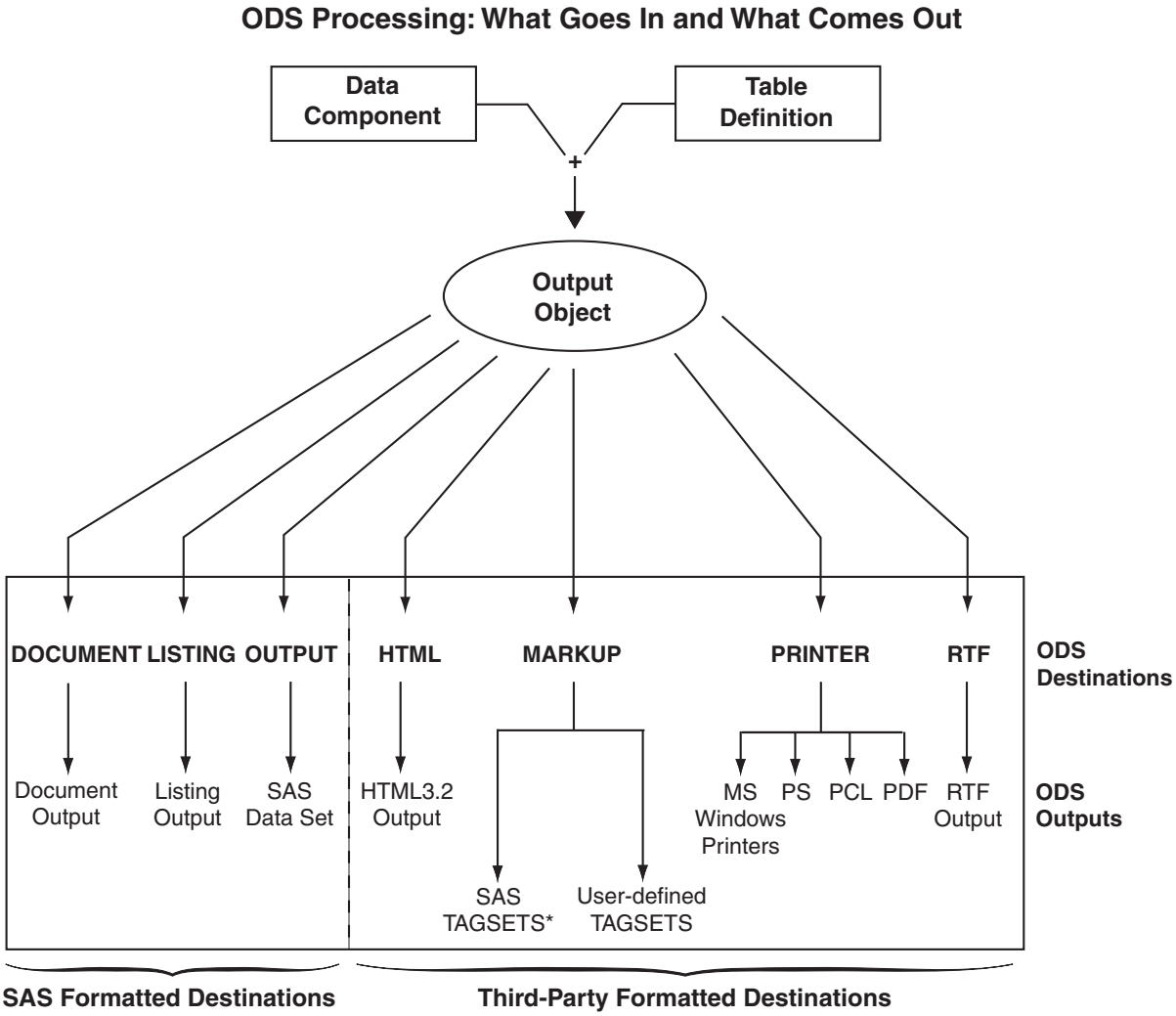


Table 2.1 * List of Tagsets That SAS Supplies and Supports

CHTML	CSV	CSVALL	CSVBYLINE
DEFAULT	DOCBOOK	EXCELXP	HTML4
HTMLCSS	HTMLPANEL	IMODE	MSOFFICE2K
PHTML	PYX	RTF	SASREPORT
WML	WMLOLIST	XHTML	

Table 2.2 * Additional Diagnostic Tagsets that SAS Supports

EVENT_MAP	NAMEDHTML	SHORT_MAP	STYLE_DISPLAY
STYLE_POPUP	TEXT_MAP	TPL_STYLE_LIST	TPL_SYLE_MAP

Note: There are also preproduction tagsets. These tagsets can be found at **`http://support.sas.com`** and are not yet supported by SAS. △

Features of ODS

ODS is designed to overcome the limitations of traditional SAS output and to make it easy to access and create the new formatting options. ODS provides a method of delivering output in a variety of formats, and makes the formatted output easy to access.

Important features of ODS include the following:

- ODS combines raw data with one or more table templates to produce one or more output objects. These objects can be sent to any or all ODS destinations. You control the specific type of output from ODS by selecting an ODS destination. The currently available ODS destinations can produce the following types of output:
  - traditional monospace output
  - an output data set
  - an ODS document that contains a hierarchy file of the output objects
  - output that is formatted for a high-resolution printer such as PostScript and PDF
  - output that is formatted in various markup languages such as HTML
  - RTF output that is formatted for use with Microsoft Word
- ODS provides table templates that define the structure of the output from SAS procedures and from the DATA step. You can customize the output by modifying these templates, or by creating your own.
- ODS provides a way for you to choose individual output objects to send to ODS destinations. For example, PROC UNIVARIATE produces five output objects. You can easily create HTML output, an output data set, traditional listing output, or printer output from any or all of these output objects. You can send different output objects to different destinations.
- In the SAS windowing environment, ODS stores a link to each output object in the Results folder in the Results window.
- Because formatting is now centralized in ODS, the addition of a new ODS destination does not affect any procedures or the DATA step. As future destinations are added to ODS, they will automatically become available to the DATA step and all procedures that support ODS.
- With ODS, you can produce output for numerous destinations from a single source, but you do not need to maintain separate sources for each destination. This feature saves you time and system resources by enabling you to produce multiple kinds of output with a single run of your procedure or data query.

Understanding ODS Destinations

Overview of ODS Destination Categories

ODS enables you to produce SAS procedure and DATA step output to many different destinations. ODS destinations are organized into two categories.

SAS Formatted destinations	produce output that is controlled and interpreted by SAS, such as a SAS data set, SAS output listing, or an ODS document.
Third-Party Formatted destinations	produce output which enables you to apply styles, markup languages, or enables you to print to physical printers using page description languages. For example, you can produce output in

PostScript, HTML, XML, or a style or markup language that you created.

The following table lists the ODS destination categories, the destination that each category includes, and the formatted output that results from each destination.

Table 2.3 Destination Category Table

Category	Destinations	Results
SAS Formatted	DOCUMENT	ODS document
	LISTING	SAS output listing
	OUTPUT	SAS data set
Third-Party Formatted	HTML	HTML file for online viewing
	MARKUP	Markup language tagsets
	PRINTER	Printable output in one of three different formats: PCL, PDF, or PS (PostScript)
	RTF	Output written in Rich Text Format for use with Microsoft Word 2000

As future destinations are added to ODS, they automatically will become available to the DATA step and to all procedures that support ODS.

Definition of Destination-Independent Input

Destination-independent input means that one destination can support a feature even though another destination does not support it. In this case, the request is ignored by the destination that does not support it. Otherwise, ODS would support a small subset of features that are common to all destinations. If this was true, then it would be difficult to move your reports from one output format to another output format. ODS provides many output formatting options, so that you can use the appropriate format for the output that you want. It is best to use the appropriate destination suited for your purpose.

The SAS Formatted Destinations

The SAS Formatted destinations create SAS entities such as a SAS data set, a SAS output listing, or an ODS document. The statements in the ODS SAS Formatted category create the SAS entities.

The three SAS Formatted destinations are as follows:

- *DOCUMENT Destination*

  The DOCUMENT destination enables you to restructure, navigate, and replay your data in different ways and to different destinations as you like without needing to rerun your analysis or repeat your database query. The DOCUMENT destination makes your entire output stream available in "raw" form and accessible to you to customize. The output is kept in the original internal representation as a data component plus a table template. When the output is in a DOCUMENT form, it is possible to rearrange, restructure, and reformat without rerunning your analysis. Unlike other ODS destinations, the DOCUMENT

destination has a GUI interface. However, everything that you can do through the GUI, you can also do with batch commands using the ODS DOCUMENT statement and the DOCUMENT procedure.

Before SAS 9, each procedure or DATA step produced output that was sent to each destination that you specified. While you could always send your output to as many destinations as you wanted, you needed to rerun your procedure or data query if you decided to use a destination that you had not originally designated. The DOCUMENT destination eliminates the need to rerun procedures or repeat data queries by enabling you to store your output objects and replay them to different destinations.

- *LISTING Destination*

  The LISTING destination produces output that looks the same as the traditional SAS output. The LISTING destination is the default destination that opens when you start your SAS session. Thus ODS is always being used, even when you do not explicitly invoke ODS.

  The LISTING destination enables you to produce traditional SAS output with the same look and presentation as it had in previous versions of SAS.

  Because most procedures share some of the same table templates, the output is more consistent. For example, if you have two different procedures producing an ANOVA table, they will both produce it in the same way because each procedure uses the same template to describe the table. However, there are three procedures that do not use a default table template to produce their output: PRINT procedure, REPORT procedure, and TABULATE procedure procedure's n-way tables. These procedures use the structure that you specified in your program code to define their tables.

- *OUTPUT Destination*

  The OUTPUT destination produces SAS output data sets. Because ODS already knows the logical structure of the data and its native form, ODS can output a SAS data set that represents exactly the same resulting data set that the procedure worked with internally. The output data sets can be used for further analysis, or for sophisticated reports in which you want to combine similar statistics across different data sets into a single table. You can easily access and process your output data sets using all of the SAS data set features. For example, you can access your output data using variable names and perform WHERE-expression processing just as you would process data from any other SAS data set.

The Third-Party Formatted Destinations

The Third-Party Formatted destinations enable you to apply styles to the output objects that are used by applications other than SAS. For example, these destinations support attributes such as "font" and "color."

Note: For a list of style attributes and valid values, see the style attributes table in "Style Attributes and Their Values" on page 498. △

The four categories of Third-Party Formatted destinations are as follows:

- *HTML (Hypertext Markup Language)*

  The HTML destination produces HTML 4.0 output that contains embedded style sheets. You can, however, produce HTML 3.2 output using the HTML3 statement.

  The HTML destination can create some or all of the following:

  - an HTML file (called the *body file*) that contains the results from the procedure

- a table of contents that links to the body file
- a table of pages that links to the body file
- a frame that displays the table of contents, the table of pages, and the body file

The body file is required with all ODS HTML output. If you do not want to link to your output, then you do not have to create a table of contents, a table of pages, or a frame file. However, if your output is very large, you might want to create a table of contents and a table of pages for easier reading and transversing through your file.

The HTML destination is intended only for online use, not for printing. To print hard-copies of the output objects, use the PRINTER destination.

- *Markup Languages (markup) Family*

  The same way as table templates describe table layout and style attributes describe the output style, *tagsets* describe how to produce a markup language output. You can use a tagset that SAS supplies or you can create your own tagset using the TEMPLATE procedure. Similar to table templates and style attributes, tagsets enable you to modify your markup language output. For example, you can specify each variety of XML as a new tagset. SAS supplies you with a collection of XML tagsets and enables you to produce a customized variety of XML.

  The important point is that you can implement either a tagset that SAS supplies or a customized tagset that you created. You do not have to wait for the next release of SAS. The additional capability to modify and create your own tagsets by using PROC TEMPLATE gives you greater flexibility in customizing your output.

  Because the MARKUP destination is so flexible, you can use either the SAS tagsets or a tagset that you created. For a complete list of the markup language tagsets that SAS supplies, see the section on listing tagset names in "ODS MARKUP Statement" on page 147. To learn how to define your own tagsets, see the section on methods to create your own tagsets in Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795.

  The MARKUP destination cannot replace ODS PRINTER or ODS RTF destinations because it cannot do text measurement. Therefore, it cannot produce output for a page description language or a hybrid language like RTF, which requires all of the text to be measured and placed at a specific position on the page.

  However, SAS 9.2 introduces a measured markup destination that is based on the traditional markup and traditional page layout destinations. The first production tagset for this destination is for RTF. Others are planned. The primary distinction of this tagset is that SAS can determine where page breaks occur in a markup language implementation. Refer to the "ODS TAGSETS.RTF Statement" on page 286 for specific information.

- *Printer Family*

  The PRINTER destination produces output for

  - printing to physical printers such as Windows printers under Windows, PCL, and PostScript printers on other operating systems
  - producing portable PostScript, PCL, and PDF files

  The PRINTER destinations produce ODS output that contain page description languages: they describe precise positions where each line of text, each rule, and each graphical element are to be placed on the page. In general, you cannot edit or alter these formats. Therefore, the output from ODS PRINTER is intended to be the final form of the report.

- *Rich Text Format (RTF)*

  RTF produces output for Microsoft Word. Other applications can read RTF files, but the RTF output might not work successfully with them.

  The RTF destination enables you to view and edit the RTF output. ODS does not define the vertical measurement, which means that SAS does not determine the optimal place to position each item on the page. For example, page breaks are not always fixed, so you do not want your RTF output tables to split at inappropriate places when you edit your text. Your tables can remain whole and intact on one page or they can have logical breaks where you specify.

  Because Microsoft Word needs to know the widths of table columns and it cannot adjust tables if they are too wide for the page, ODS measures the width of the text and tables (horizontal measurement). Therefore SAS can set all of the column widths properly and divide the table into panels if it is too wide to fit on a single page.

  In short, when SAS produces RTF output for input to Microsoft Word, it determines the horizontal measurement. Microsoft Word controls the vertical measurement. Because Microsoft Word can determine how much space is on the page, your tables display consistently even after you make changes to your RTF file.

  However, when you use measured RTF, which is implemented when you use the ODS TAGSETS.RTF statement, you can specify how and where page breaks occur. You can also specify when to place titles and footnotes into the body of a page. SAS becomes responsible for the implicit page breaks instead of Microsoft Word. Refer to the "ODS TAGSETS.RTF Statement" on page 286 for specific information.

Controlling the Formatting Features of Third-Party Formats

All of the formatting features that control the appearance of the third-party formatted destinations beyond what the LISTING destination can do are controlled by two mechanisms:

- ODS statement options
- ODS style attributes

The ODS statement options control three features:

1. features that are specific to a given destination, such as style sheets for HTML
2. features that are global to the document, such as AUTHOR and table of contents generation
3. features that we expect programmers to change on each document, such as the output filename

The ODS style attributes control the way that individual elements are created. Attributes are aspects of a given style, such as type face, weight, font size, and color. The values of the attributes collectively determine the appearance of each part of the document to which the style is applied. With style attributes, it is unnecessary to insert destination-specific code (such as raw HTML) into the document. Each output destination will interpret the attributes that are necessary to generate the presentation of the document. Because not all destinations are the same, not all attributes can be interpreted by all destinations. Style attributes that are incompatible with a selected destination are ignored. For example, PostScript does not support active links, so the URL= attribute is ignored when producing PostScript output.

ODS Destinations and System Resources

ODS destinations can be open or closed. You open and close a destination with the appropriate ODS statement. When a destination is open, ODS sends the output objects to it. An open destination uses system resources even if you use the selection and exclusion features of ODS to select or exclude all objects from the destination. Therefore, to conserve resources, close unnecessary destinations. For more information about using each destination, see the topic on ODS statements in Chapter 5, "Dictionary of ODS Language Statements," on page 67.

By default, the LISTING destination is open and all other destinations are closed. Consequently, if you do nothing, your SAS programs run and produce listing output looking just as they did in previous releases of SAS before ODS was available.

Understanding Table Templates, Table Elements, and Table Attributes

A *table template* describes how to generate the output for a tabular output object. (Most ODS output is tabular.) A table template determines the order of column headings and the order of variables, as well the overall look of the output object that uses it. For information about customizing the table template, see the topic on the TEMPLATE procedure in Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593.

In addition to the parts of the table template that order the headers and columns, each table template contains or references *table elements*. A table element is a collection of table attributes that apply to a particular header, footer, or column. Typically, a *table attribute* specifies something about the data rather than about its presentation. For example, FORMAT specifies the SAS format, such as the number of decimal places. However, some table attributes describe presentation aspects of the data, such as how many blank characters to place between columns.

Note: The attributes of table templates that control the presentation of the data have no effect on output objects that go to the LISTING or OUTPUT destination. However, the attributes that control the structure of the table and the data values do affect listing output. △

For information on table attributes, see the section on table attributes in Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593.

Understanding Styles, Style Elements, and Style Attributes

To customize the output at the level of your entire output stream in a SAS session, you specify a style. A *style* describes how to generate the presentation aspects (color, font face, font size, and so on) of the entire SAS output. A style determines the overall look of the documents that use it.

Each style consists of *style elements*. A style element is a collection of style attributes that apply to a particular part of the output. For example, a style element might contain instructions for the presentation of column headings, or for the presentation of the data inside the cells. Style elements might also specify default colors and fonts for output that uses the style.

Each *style attribute* specifies a value for one aspect of the presentation. For example, the BACKGROUND= attribute specifies the color for the background of an HTML table

or for a colored table in printed output. The FONTSTYLE= attribute specifies whether to use a Roman or an italic font. For information on style attributes, see the section on style attributes in Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.

Note: Because styles control the presentation of the data, they have no effect on output objects that go to the LISTING or OUTPUT destination. △

Styles That Are Shipped with SAS Software

Base SAS software is shipped with many styles. To see a list of these styles, view them in the SAS Explorer Window, use the TEMPLATE procedure, or use the SQL procedure.

- *SAS Explorer Window:*

  To display a list of the available styles using the SAS Explorer Window, follow these steps:

  1. From any window in an interactive SAS session, select **View ▶ Results**
  2. In the Results window, select **View ▶ Templates**
  3. In the Templates window, select and open **`Sashelp.tmplmst`**.
  4. Select and open the **`Styles`** folder, which contains a list of available styles. If you want to view the underlying SAS code for a style, then select the style and open it.

     Operating Environment Information: For information on navigating in the Explorer window without a mouse, see the section on "Window Controls and General Navigation" in the SAS documentation for your operating environment. △

- *TEMPLATE Procedure:*

  You can also display a list of the available styles by submitting the following PROC TEMPLATE statements:

```
proc template;
   list styles;
run;
```

- *SQL Procedure:*

  You can also display a list of the available styles by submitting the following PROC SQL statements:

```
proc sql;
select * from dictionary.styles;
quit;
```

For more information on how ODS destinations use styles and how you can customize styles, see the "DEFINE STYLE Statement" on page 490.

Using Styles with Base SAS Procedures

- Most Base SAS Procedures

  Most Base SAS procedures that support ODS use one or more table templates to produce output objects. These table templates include templates for table elements: columns, headers, and footers. Each table element can specify the use of

one or more style elements for various parts of the output. These style elements cannot be specified within the syntax of the procedure, but you can use customized styles for the ODS destinations that you use. For more information about customizing tables and styles, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.

- The PRINT, REPORT, and TABULATE Procedures

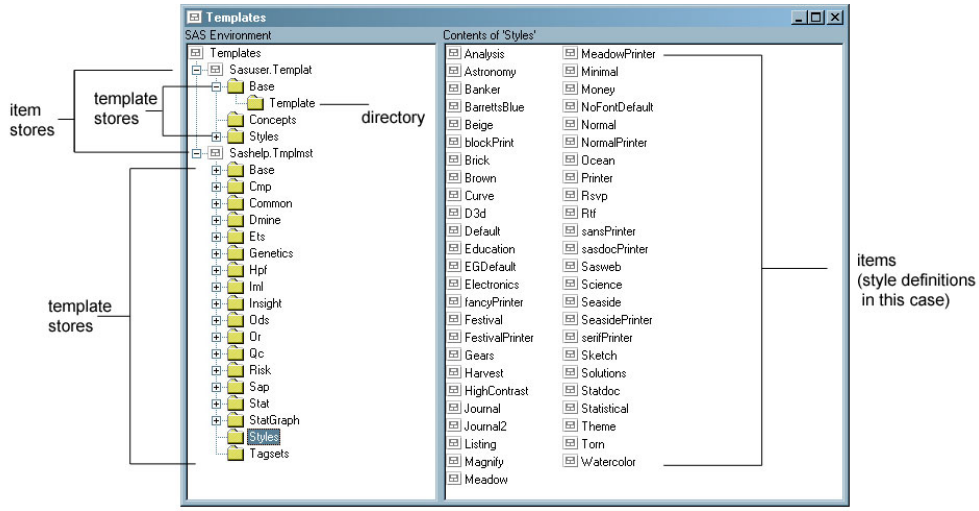
  The PRINT, REPORT, and TABULATE procedures provide a way for you to access table elements from the procedure step itself. Accessing the table elements enables you to do such things as specify background colors for specific cells, change the font face for column headings, and more. The PRINT, REPORT, and TABULATE procedures provide a way for you to customize the markup language and printed output directly from the procedure statements that create the report. For more information about customizing the styles for these procedures, see the *Base SAS Procedures Guide*.

Understanding Item Stores, Template Stores, and Directories

A template store is an item store which stores items that were created by the TEMPLATE procedure. Items that SAS provides are in the item store SASHELP.TMPLMST. Compiled templates are stored physically in the SASUSER.TEMPLAT item store by default. However, you can store items that you create in any template store where you have write access.

A template store can contain multiple levels known as directories. When you specify a template store in the ODS PATH statement, however, you specify a two-level name that includes a libref and the name of a template store in the SAS library that the libref references.

Display 2.5 Templates Window Showing Item Stores, Template Stores, Directories, and Items

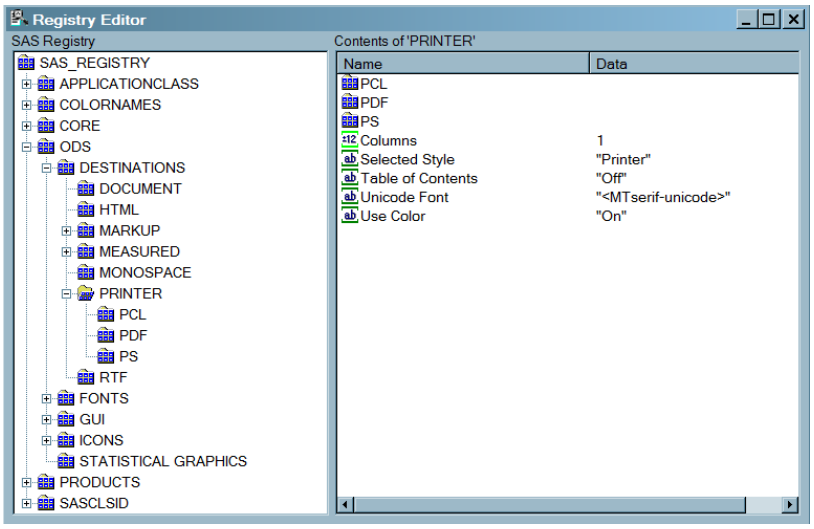


Changing SAS Registry Settings for ODS

Overview of ODS and the SAS Registry

The SAS registry is the central storage area for configuration data that ODS uses. This configuration data is stored in a hierarchical form, which works in a similar manner to the way directory-based file structures work under UNIX, Windows, VMS, and the z/OS UNIX system. However, the SAS registry uses keys and subkeys as the basis for its structure, instead of using directories and subdirectories, like similar file systems in DOS or UNIX. A key is a word or a text string that refers to a particular aspect of SAS. Each key might be a place holder without values or subkeys associated with it, or it might have many subkeys with associated values. For example, the ODS key has DESTINATIONS, GUI, ICONS, and PREFERENCES subkeys. A subkey is a key inside another key. For example, PRINTER is a subkey of the DESTINATIONS subkey.

Display 2.6 SAS Registry of ODS Subkeys



Changing Your Default HTML Version Setting

By default, the SAS registry is configured to generate HTML4 output when you specify the ODS HTML statement. To permanently change the default HTML version, you can change the setting of the HTML version in the SAS registry.

CAUTION:

If you make a mistake when you modify the SAS registry, then your system might become unstable or unusable. You will not be warned if an entry is incorrect. Incorrect entries can cause errors, and can even prevent you from bringing up a SAS session. See the section on configuring the SAS registry in *SAS Language Reference: Concepts* for more information. △

To change the default setting of the HTML version in the SAS registry:

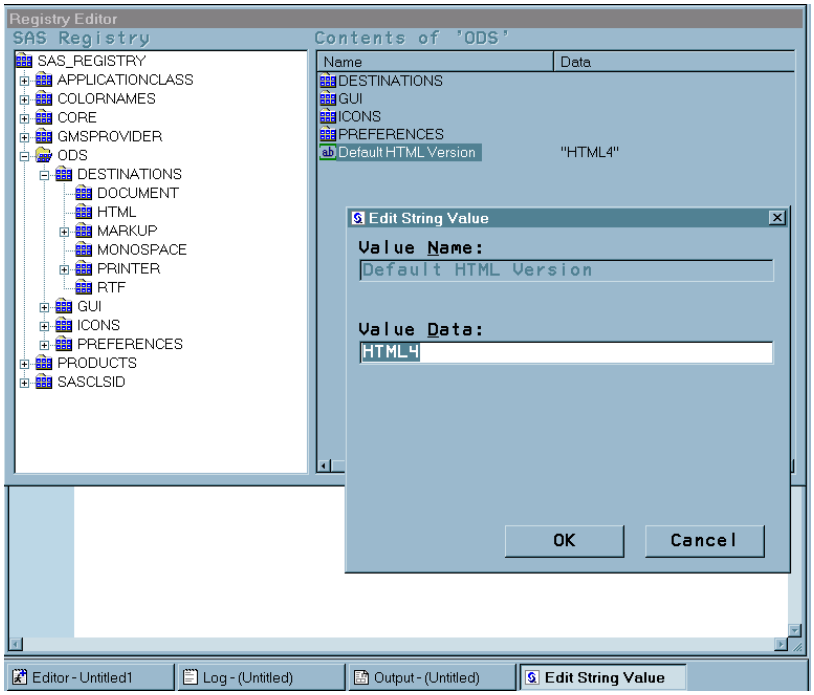
1 Select **Accessories ▶ Solutions ▶ Registry Editor** *or* issue the command **`REGEDIT`**.

2 Select **ODS ▶ Default HMTL Version**.

3 Select **Edit ▶ Modify** *or*
click the right mouse button and select **MODIFY**. The Edit String Value window appears.

4 Type the HTML version in the **Value Data** text box and select **OK**.

Display 2.7 SAS Registry Showing HTML Version Setting



Changing ODS Destination Default Settings

ODS destination subkeys are stored in the SAS registry. To change the values for these destinations subkeys:

1 Select **ODS ▶ Destinations**.

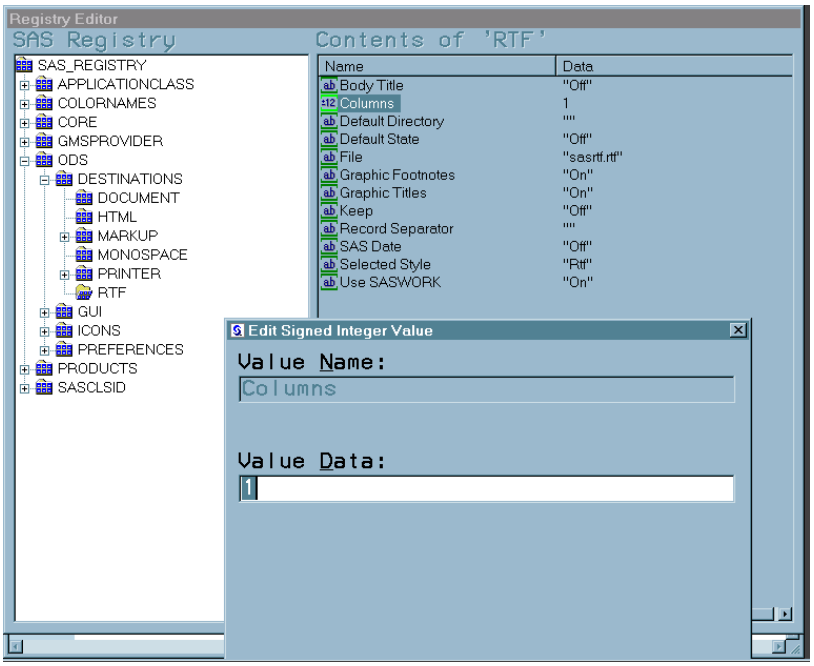
2 Select a destination subkey.

3 Select a subkey in the **Contents of** pane.

4 Select **Edit ▶ Modify** *or*
click the right mouse button and select **MODIFY**.

5 Type the value into the **Value Data** text box in the Edit Value String or Edit Signed Integer Value dialog box and select **OK**.

Display 2.8 Registry Editor Window



Customized ODS Output

SAS Output

By default, ODS output is formatted according to instructions that a PROC step or DATA step defines. However, ODS provides ways for you to customize the output. You can customize the output for an entire SAS job, or you can customize the output for a single output object.

Selection and Exclusion Lists

For each ODS destination, ODS maintains either a selection list or an exclusion list of output objects. You can use the default output objects selected or excluded for each destination or you can specify which output object you want to produce by selecting or excluding them from a list.

A selection list is a list of output objects that are sent to an ODS destination. An exclusion list is a list of output objects that are excluded from an ODS destination. ODS also maintains an overall selection or exclusion list of output objects. By checking the destination-specific lists and the overall list, ODS determines what output objects to produce. These lists can be modified by using the ODS SELECT statement and the ODS EXCLUDE statement.

You can view the contents of the exclusion and selection lists by using the ODS SHOW statement. The contents information is written to the SAS log.

EXCLUDE ALL is the default setting for the ODS OUTPUT destination. SELECT ALL is the default setting for all other destinations. To change the default selection and

exclusion lists, use the ODS SELECT or ODS EXCLUDE statements or use the exclude and select actions that are available for some of the ODS statements. However, to set the exclusion list for the OUTPUT destination to something other than the default, use the "ODS OUTPUT Statement" on page 184. For a list of ODS Output destinations and explanations of each, see "Understanding ODS Destinations" on page 24.

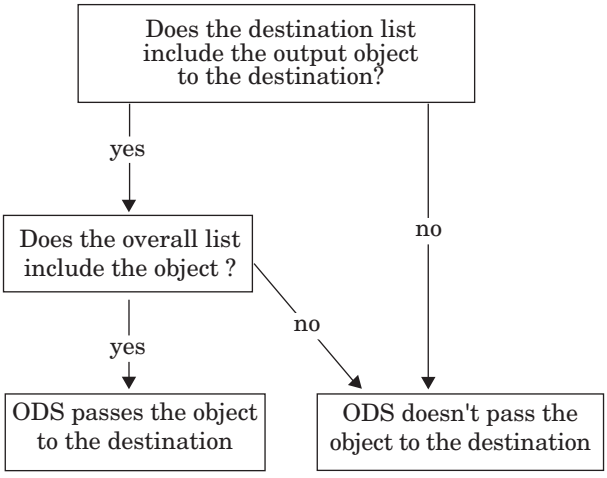
In order to view output objects that are selected or excluded from your program, use the ODS TRACE statement. The ODS TRACE statement prints the output objects that are selected and excluded and puts the information in a trace record that is output in the SAS log. The trace provides the path, the label, and other information about output objects that are selected and excluded. For complete documentation about viewing and selecting output objects, see the "ODS SELECT Statement" on page 264, the "ODS EXCLUDE Statement" on page 110, and the "ODS TRACE Statement" on page 317.

How ODS Determines the Destinations for an Output Object

As each output object is produced, ODS uses the selection and exclusion lists to determine which destination or destinations the output object will be sent to. Figure Figure 2.2 on page 35 illustrates this process:

Figure 2.2 Directing an Output Object to a Destination

For each destination, ODS first asks if the list for that destination includes the object. If it does not, ODS does not send the output object to that destination. If the list for that destination does include the object, ODS reads the overall list. If the overall list includes the object, ODS sends it to the destination. If the overall list does not include the object, ODS does not send it to the destination.



Note: Although you can maintain a selection list for one destination and an exclusion list for another, it is easier to understand the results if you maintain the same types of lists for all the destinations where you route output. △

Customized Output for an Output Object

For a procedure, the name of the table template that is used for an output object comes from the procedure code. The DATA step uses a default table template unless

you specify an alternative with the TEMPLATE= suboption in the ODS option in the FILE statement. For more information, see the section on the TEMPLATE= suboption in "FILE Statement for ODS" on page 68.

To find out which table templates a procedure or the DATA step uses for the output objects, you must look at a trace record. To produce a trace record in your SAS log, submit the following SAS statements:

```
ods trace on;
your-proc-or-DATA-step
ods trace off;
```

Remember that not all procedures use table templates. If you produce a trace record for one of these procedures, no template appears in the trace record. Conversely, some procedures use multiple table templates to produce their output. More than one template appears in the trace record produced in the log.

For a detailed explanation of the trace record, see the "ODS TRACE Statement" on page 317.

You can use PROC TEMPLATE to modify an entire table template. When a procedure or DATA step uses a table template, it uses the elements that are defined or referenced in its table template. In general, you cannot directly specify a table element for your procedure or DATA step to use without modifying the template itself.

Note: Three Base SAS procedures, PROC PRINT, PROC REPORT, and PROC TABULATE, do provide a way for you to access table elements from the procedure step itself. Accessing the table elements enables you to customize your report. For more information about these procedures, see the *Base SAS Procedures Guide*. △

Customizing Titles and Footnotes

You can use the global TITLE and FOOTNOTE statements to enhance the readability of any report. These statements have associated options that enable you to customize the style of the titles and footnotes when they are used with ODS. Because these options control only the presentation of the titles and footnotes, they have no effect on objects that go to the LISTING or OUTPUT destination. Examples of these style options are: BOLD, COLOR=, and FONT=. For a complete list of style options, detailed information about the style options, and example code, refer to the TITLE statement and the FOOTNOTE statement in the *SAS Language Reference: Dictionary*.

When used with SAS/GRAPH, you can choose whether to render the titles and footnotes as part of the body of the document or as part of the graphics image. Where the titles and footnotes are rendered determines how you control the font, size, and color of the titles and footnotes text. For details on this ODS and SAS/GRAPH interaction, refer to Controlling Titles and Footnotes with ODS Output in *SAS/GRAPH: Reference*.

For information on titles and footnotes rendered with and without using the graphics option USEGOPT, refer to "ODS USEGOPT Statement" on page 322.

Securing ODS Generated PDF Files

You can use the "ODS PRINTER Statement" on page 218 or the "ODS PDF Statement" on page 210 to generate PDF output. When these PDF files are not password protected, any user can use Acrobat to view and edit the PDF files. However, SAS system options can restrict or allow users' ability to access, assemble, copy, or modify the ODS PDF files. Other SAS system options control whether the user can fill in forms and set the print resolution. The following SAS system options are documented in *SAS Language Reference: Dictionary*.

Table 2.4 PDF System Options

Task	System Option
Specifies whether text and graphics from PDF documents can be read by screen readers for the visually impaired	PDFACCESS | NOPDFACCESS
Controls whether PDF documents can be assembled	PDFASSEMBLY | NOPDFASSEMBLY
Controls whether PDF document comments can be modified	PDFCOMMENT | NOPDFCOMMENT
Controls whether the contents of a PDF document can be changed	PDFCONTENT | NOPDFCONTENT
Controls whether text and graphics from a PDF document can be copied	PDFCOPY | NOPDFCOPY
Controls whether PDF forms can be filled in	PDFFILLIN | NOPDFFILLIN
Specifies the password to use to open a PDF document and the password used by a PDF document owner	PDFPASSWORD
Controls the resolution used to print the PDF document	PDFPRINT
Controls the printing permissions for PDF documents	PDFSECURITY

Note: The SAS/SECURE SSL software is included in the SAS installation software only for countries that allow the importation of encryption software. △

Summary of ODS

In the past, the term "output" has generally referred to the outcome of a SAS procedure and DATA step. With the advent of the Output Delivery System, output takes on a much broader meaning. ODS optimizes output from SAS procedures and the DATA step. ODS provides a wide range of formatting options and greater flexibility in generating, storing, and reproducing SAS output.

Important features of ODS include the following:

- ODS combines raw data with one or more table templates to produce one or more *output objects*. An output object tells ODS how to format the results of a procedure or DATA step.
- ODS provides table templates that define the structure of the output from SAS procedures and from the DATA step. You can modify these templates or create your own templates to customize your output.
- ODS provides a way for you to choose individual output objects to send to ODS destinations.
- ODS stores a link to each output object in the Results folder for easy retrieval and access.
- As future destinations are added to ODS, these destinations automatically become available to the DATA step and all procedures that support ODS.

One of the main goals of ODS is to enable you to produce output for numerous destinations from a single source, without requiring separate sources for each destination. ODS supports many destinations:

DOCUMENT
: enables you to capture output objects from a single run of the analysis and to produce multiple reports in various formats whenever you want without rerunning your SAS programs.

LISTING
: produces output that looks the same as the traditional SAS output.

HTML
: produces output for online viewing.

MARKUP
: produces output for markup language tagsets.

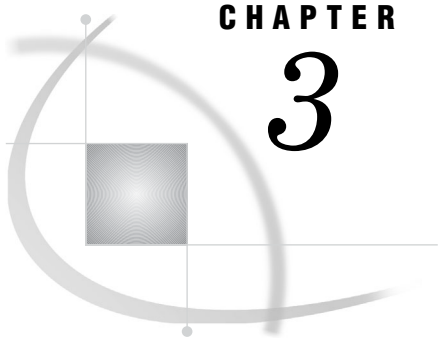
MEASURED MARKUP
: produces output for page-oriented markup languages.

OUTPUT
: produces SAS output data sets, thereby eliminating the need to parse PROC PRINTTO output.

PRINTER
: produces presentation-ready printed reports.

RTF
: produces output suitable for Microsoft Word reports.

By default, ODS output is formatted according to instructions that the procedure or DATA step defines. However, ODS provides ways for you to customize the presentation of your output. You can customize the presentation of your SAS output, or you can customize the look of a single output object. ODS gives you greater flexibility in generating, storing, and reproducing SAS procedure and DATA step output with a wide range of formatting options.



CHAPTER

3

Output Delivery System and the DATA Step

Using ODS with the DATA Step

If you are writing DATA step reports now, you are already using ODS. Simple listing output, the traditional DATA step output, is routed though ODS by default. For over 20 years, SAS users have been able to create highly customized reports as simple listing output, which uses a monospace typefont. With the advent of ODS, however, you have a broad range of choices for printing your customized DATA step reports:

- You can produce DATA step reports in many different formats, such as HTML, RTF, PS (PostScript), or PDF.
- You can create the report in multiple formats at the same time.
- You can also produce the report in different formats at a later time without rerunning the DATA step.

To take advantage of these enhanced reporting capabilities, you can combine DATA step programming with the formatting capabilities of ODS.

To create PDF output, for example, start with the DATA steps tools that you are already familiar with:

- the DATA _NULL_ statement

- the FILE statement
- the PUT statement

Then, add a few simple ODS statements and options. In addition, you can choose from several ODS formatting statements to format the output in other presentation styles, such as HTML, RTF, and PS. For more information on ODS statements, see Chapter 5, "Dictionary of ODS Language Statements," on page 67.

How ODS Works with the DATA Step

Here are the basic steps for using ODS in conjunction with the DATA step to produce reports with enhanced formatting:

Table 3.1 Steps to Producing Enhanced ODS Output With the DATA Step

Steps	Tools	Comments
Specify formatting for your output	ODS formatting statements can specify formats such as listing, HTML, RTF, PS, and PDF.	You can also produce output in multiple formats at the same time by specifying more than one format. Note: If you want only the simple default listing output, then you don't need the ODS statement.
Specify structure	The ODS option in the FILE statement lists the variables and their order in the output.	Additional suboptions give you even more control over the resulting structure.
Connect the data to the template	The FILE PRINT ODS statement creates an output object by binding a data component to a table definition (template).	You can specify other details by using various ODS suboptions in the FILE PRINT ODS statement.
Output data	The PUT statement writes variable values to the data component.	A simple way to output all variables is to use PUT _ODS_.

First, use ODS statements to specify how you want ODS to format your output, for example, as HTML, RTF or PDF. Then, in the DATA step, use the FILE PRINT ODS and PUT statements, with appropriate ODS-specific suboptions, to produce your report.

The PUT statement writes variable values, and the FILE PRINT ODS statement directs the output.* You can use ODS to produce the same output in multiple formats, and to produce output at a later time in a different format, without rerunning the DATA step.

You control the formatting that is applied to your reports by using the ODS formatting statements. They control the opening and closing of ODS destinations, which apply formatting to the output objects that you create with ODS and the DATA step.

Here is a list of topics, with sources for additional information.

* If you do not specify a FILE statement, then the PUT statement writes to the SAS log by default. If you use multiple PUT and FILE statements, then in addition to creating ODS-enhanced output, you can write to the log, to the regular DATA step output buffer, or to another external file in the same DATA step.

Table 3.2 Where to Find More Information on How to Use ODS in the DATA Step

Topic	Where to learn more
ODS formatting statements	Chapter 5, "Dictionary of ODS Language Statements," on page 67
ODS destinations	"Understanding ODS Destinations" on page 24
How ODS works	"Overview of How ODS Works" on page 22

Syntax for ODS Enhanced Features in a DATA Step

Restriction:
To use the DATA step and ODS to produce output that contains more enhanced formatting features than the default listing output, you must use both the FILE PRINT ODS statement and the PUT statement.

See:
"FILE Statement for ODS" on page 68 and "PUT Statement for ODS" on page 81

FILE PRINT ODS<=(*ODS-suboption(s)*)><*options*>;

PUT <*specification(s)*> <_ODS_ <@|@@>> ;

Examples

Example 1: Creating a Report with the DATA Step and the Default Table Definition

ODS features:

FILE PRINT ODS statement

PUT _ODS_ statement

ODS destinations:

LISTING

This example uses the DATA step and ODS to create a listing report. It uses the default table definition (template) for the DATA step and writes an output object to the LISTING destination (the default).

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The PAGENO= option specifies the starting page number. The LINESIZE= option specifies the output line length, and the PAGESIZE= option specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=64 pagesize=60;
```

Specify a title. The TITLE statement specifies a title for the output.

```
title 'Leading Grain Producers';
```

Create a user-defined format. PROC FORMAT creates the format $CNTRY. for the variable COUNTRY.

```
proc format;
   value $cntry 'BRZ'='Brazil'
                'CHN'='China'
                'IND'='India'
                'INS'='Indonesia'
                'USA'='United States';
run;
```

Begin a DATA step that does not create an output data set. Using _NULL_ saves computer resources because it prevents the DATA step from creating an output data set.

```
data _null_;
```

Define variables, assign lengths and formats, read a record, and assign values to four variables. The LENGTH statement defines a length that is shorter than the default to two character variables. The FORMAT statement assigns a user-defined format to the variable COUNTRY. The LABEL statement assigns a label to the variable TYPE. The INPUT statement reads a record from the datalines and assigns a value to four variables.

```
   length Country $ 3 Type $ 5;
   format country $cntry.;
   label type='Grain';
   input Year country $ type $ Kilotons;
```

Use the default table definition (template) to create simple listing output. The combination of the fileref PRINT and the ODS option in the FILE statement routes the DATA step output to ODS. The only open ODS destination is the LISTING destination, which is open by default when you begin your SAS session. Because no suboptions are specified, ODS uses the default DATA step table definition (template). This FILE PRINT ODS statement creates an output object and binds it to the default template.

```
   file print ods;
```

Write the variables to the data component. The _ODS_ option in the PUT statement writes every variable to the buffer that the PUT statement writes to the data component. Because no formats or labels are specified for individual columns, ODS uses the defaults.

```
   put _ods_;
```

The data provide information on the amounts of wheat, rice, and corn that five leading grain-producing nations produced during 1995 and 1996.

```
   datalines;
1995 BRZ  Wheat    1516
1995 BRZ  Rice     11236
1995 BRZ  Corn     36276
1995 CHN  Wheat    102207
```

```
1995 CHN  Rice     185226
1995 CHN  Corn     112331
1995 IND  Wheat    63007
1995 IND  Rice     122372
1995 IND  Corn     9800
1995 INS  Wheat    .
1995 INS  Rice     49860
1995 INS  Corn     8223
1995 USA  Wheat    59494
1995 USA  Rice     7888
1995 USA  Corn     187300
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 CHN  Wheat    109000
1996 CHN  Rice     190100
1996 CHN  Corn     119350
1996 IND  Wheat    62620
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 INS  Wheat    .
1996 INS  Rice     51165
1996 INS  Corn     8925
1996 USA  Wheat    62099
1996 USA  Rice     7771
1996 USA  Corn     236064
;
```

Listing Output

Output 3.1 Listing Output Created with the Default DATA Step Table Definition

The default table definition produces a column for each variable in the DATA step. The order of the columns is determined by their order in the program data vector. Because no attributes are specified for individual columns, ODS uses the default column headings and formats.

Leading Grain Producers 1

Country	Grain	Year	Kilotons
Brazil	Wheat	1995	1516
Brazil	Rice	1995	11236
Brazil	Corn	1995	36276
China	Wheat	1995	102207
China	Rice	1995	185226
China	Corn	1995	112331
India	Wheat	1995	63007
India	Rice	1995	122372
India	Corn	1995	9800
Indonesia	Wheat	1995	.
Indonesia	Rice	1995	49860
Indonesia	Corn	1995	8223
United States	Wheat	1995	59494
United States	Rice	1995	7888
United States	Corn	1995	187300
Brazil	Wheat	1996	3302
Brazil	Rice	1996	10035
Brazil	Corn	1996	31975
China	Wheat	1996	109000
China	Rice	1996	190100
China	Corn	1996	119350
India	Wheat	1996	62620
India	Rice	1996	120012
India	Corn	1996	8660
Indonesia	Wheat	1996	.
Indonesia	Rice	1996	51165
Indonesia	Corn	1996	8925
United States	Wheat	1996	62099
United States	Rice	1996	7771
United States	Corn	1996	236064

Example 2: Producing ODS Output That Contains Selected Variables

ODS features:

- FILE PRINT ODS statement:
  - VARIABLES= suboption
- ODS HTML statement:
  - BODY= option
    - URL= suboption
- PUT _ODS_ statement

ODS destinations:

- HTML
- LISTING

Format:
See “Creating the $CNTRY Format” on page 869.

This example selects variables to include in the output. The resulting output is produced in two formats, listing and HTML. The listing output is produced by default, and the HTML output is requested by the ODS HTML statement.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 903. △

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The PAGENO= option specifies the starting page number. The LINESIZE= option specifies the output line length, and the PAGESIZE= option specifies the number of lines on an output page. None of these options affects the HTML output.

```
options nodate pageno=1 linesize=64 pagesize=60;
```

Specify that you want ODS to create HTML output and store it in the specified file. The ODS HTML statement opens the HTML destination; any procedure or DATA step output created will be routed to this destination (and any others that are open) and will, therefore, format the output in HTML. The BODY= option sends all output objects to the HTML file that you specify. Some browsers require an extension of HTM or HTML on the filename.

```
ods html body='your-html-file.html';
```

Specify the titles. The TITLE statements provide titles for the output.

```
title 'Leading Grain Producers';
title2 'for 1996';
```

Begin a DATA step that does not create an output data set. Using _NULL_ saves computer resources because it prevents the DATA step from creating an output data set.

```
data _null_;
```

Assign lengths other than the default to two character variables. Also assign a user defined format to one variable and a label to another. The FORMAT statement assigns a format to the variable COUNTRY. The LABEL statement assigns a label to the variable TYPE.

```
    length Country $ 3 Type $ 5;
    format country $cntry.;
    label type='Grain';
```

Read a record from the input data, assign values to four variables. Continue to process only observations that match the criterion. The INPUT statement reads a single record and assigns values to four variables. The subsetting IF statement causes the DATA step to continue to process only those observations that contain the value 1996 for YEAR.

```
    input Year country $ type $ Kilotons;
    if year=1996;
```

Send the DATA step output to whatever ODS destinations are open. Specify the variables and their order in the data component that is created. The combination of the fileref PRINT and the ODS option in the FILE statement sends the results of the DATA step to ODS. Two ODS destinations, the LISTING and the HTML destinations, are open. Because no table definition is specified, ODS uses the default DATA step definition. The VARIABLES= suboption specifies that the resulting data component will contain three columns in the order that is listed.

```
file print ods=(variables=(country
                           type
                           kilotons));
```

Write values for all variables that are specified with the VARIABLES= suboption in the FILE statement. The _ODS_ option in the PUT statement writes variable values to the data component. It writes only those variables that were specified with the VARIABLES= suboption in the FILE statement. Because no formats or labels are specified for these ODS columns, ODS uses the defaults.

```
put _ods_;
```

The data provides information on the amounts of wheat, rice, and corn that were produced by the five leading grain-producing nations during 1995 and 1996.

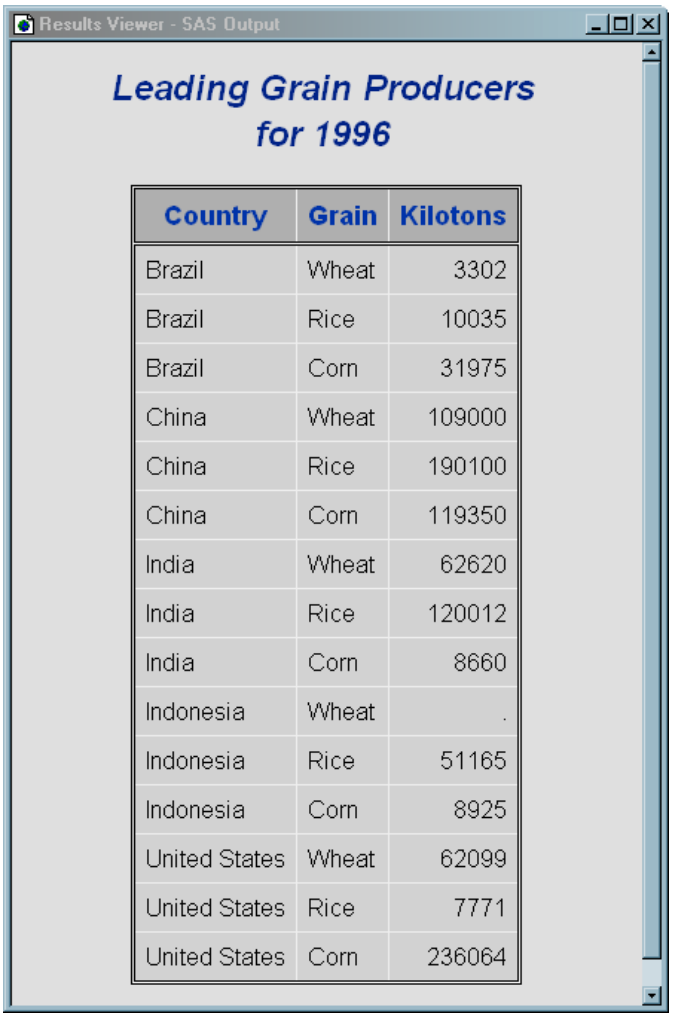
```
datalines;
1995 BRZ  Wheat    1516
1995 BRZ  Rice     11236
1995 BRZ  Corn     36276
1995 CHN  Wheat    102207
1995 CHN  Rice     185226
1995 CHN  Corn     112331
1995 IND  Wheat    63007
1995 IND  Rice     122372
1995 IND  Corn     9800
1995 INS  Wheat    .
1995 INS  Rice     49860
1995 INS  Corn     8223
1995 USA  Wheat    59494
1995 USA  Rice     7888
1995 USA  Corn     187300
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 CHN  Wheat    109000
1996 CHN  Rice     190100
1996 CHN  Corn     119350
1996 IND  Wheat    62620
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 INS  Wheat    .
1996 INS  Rice     51165
1996 INS  Corn     8925
1996 USA  Wheat    62099
1996 USA  Rice     7771
1996 USA  Corn     236064
;
```

Close the HTML destination so that you can view the output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser. Also, closing the destination prevents all subsequent ODS jobs from automatically producing HTML output.

```
ods html close;
```

HTML Output

Display 3.1 HTML Body File Produced by ODS



Leading Grain Producers for 1996

Country	Grain	Kilotons
Brazil	Wheat	3302
Brazil	Rice	10035
Brazil	Corn	31975
China	Wheat	109000
China	Rice	190100
China	Corn	119350
India	Wheat	62620
India	Rice	120012
India	Corn	8660
Indonesia	Wheat	.
Indonesia	Rice	51165
Indonesia	Corn	8925
United States	Wheat	62099
United States	Rice	7771
United States	Corn	236064

Listing Output

Output 3.2 Listing Output Produced by the LISTING Destination

```
                   Leading Grain Producers                     1
                           for 1996
               Country       Grain      Kilotons

            Brazil           Wheat           3302
            Brazil           Rice           10035
            Brazil           Corn           31975
            China            Wheat         109000
            China            Rice          190100
            China            Corn          119350
            India            Wheat          62620
            India            Rice          120012
            India            Corn            8660
            Indonesia        Wheat              .
            Indonesia        Rice           51165
            Indonesia        Corn            8925
            United States    Wheat          62099
            United States    Rice            7771
            United States    Corn          236064
```

Example 3: Assigning Attributes to Columns in ODS Output

ODS features:
- FILE PRINT ODS statement:
  - OBJECTLABEL= suboption
  - VARIABLES= suboption
    - LABEL= suboption
    - FORMAT= suboption
- PUT _ODS_ statement

ODS destinations:
- HTML
- LISTNG
- PRINTER (PS)

Format:
See “Creating the $CNTRY Format” on page 869.

This example assigns a label to the output object that it creates. It also specifies a label and a format for individual columns.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 903. △

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The PAGENO= option specifies the starting page number. The LINESIZE= option specifies the output line length, and the PAGESIZE= option specifies the number of lines on an output page. These options affect the listing output, but none of them affects the HTML output.

```
options pagesize=60 linesize=64 nodate pageno=1;
```

Specify that you want to create HTML output. Also specify where to store the HTML output: the body file, the contents file, and the frame file. The ODS HTML statement opens the HTML destination and creates HTML output. The BODY= option identifies the file that contains the HTML output. The CONTENTS= option identifies the file that contains a table of contents to the HTML output. The contents file links to the body file. The FRAME= option identifies the file that integrates the table of contents, the page contents, and the body file. If you open the frame file, you see a table of contents, a table of pages, or both, as well as the body file.

```
ods html body='your_body_file.html'
         contents='your_contents_file.html'
         frame='your_frame_file.html';
```

Specify that you want PostScript output. Also specify where to store the PostScript output. The ODS PRINTER statement opens the PRINTER destination and creates PostScript output by default. The FILE= option sends all output objects to the external file in the current directory.

```
ods printer file='your_postscript_file.ps';
```

Specify the titles. The TITLE statements provide titles for the output.

```
title 'Leading Grain Producers';
title2 'for 1996';
```

Begin a DATA step that does not create an output data set. Using _NULL_ saves computer resources because it prevents the DATA step from creating an output data set.

```
data _null_;
```

Assign lengths other than the default to two character variables. Also assign a user defined format to one variable and a label to another. The LENGTH statement assigns lengths to COUNTRY and TYPE. The FORMAT statement assigns a format to the variable COUNTRY. The LABEL statement assigns a label to the variable TYPE.

```
    length Country $ 3 Type $ 5;
    format country $cntry.;
    label type='Grain';
```

Read a record from the input data, assign values to four variables. Continue to process only observations that match the criterion. The INPUT statement reads a single record and assigns values to four variables. The subsetting IF statement causes the DATA step to continue to process only those observations that contain the value 1996 for YEAR.

```
    input Year country $ type $ Kilotons;
    if year=1996;
```

Send the DATA step output to the open destinations, specify a label for the output object, and specify the variables to write to the data component and the order in which to write them. The combination of the fileref PRINT and the ODS option in the FILE statement sends the results of the DATA step to ODS. The LISTING, the HTML, and the PRINTER destinations are open. Because no table definition is specified, ODS uses the default DATA step definition.

- The OBJECTLABEL= suboption specifies the label '1996 Grain Production' to the output object. This label appears in the Results folder and in the HTML contents file.
- The VARIABLES= suboption specifies the variables to write to the data component and the order in which to write them.
- The LABEL= suboption specifies a label for the variable TYPE. The label specified here takes precedence over the LABEL statement assignment that was made previously in the DATA step, so it is used as the column heading for TYPE.
- The FORMAT= suboption assigns a format for the variable KILOTONS.

```
file print ods= (objectlabel='1996 Grain Production'
        variables=(country
                   type(label='Type of Grain')
                   kilotons(format=comma12.))
   );
```

Write the variables to the buffer. The _ODS_ option in the PUT statement writes all of the variables that are defined to ODS (in the FILE PRINT ODS statement) to a special buffer. It uses default attributes for COUNTRY, and it uses any attributes specified in the VARIABLES= suboption for the other variables. For attributes that might be specified elsewhere in the DATA step but are not specified in VARIABLES=, it uses the defaults.

```
put _ods_;
```

The data provides information on the amounts of wheat, rice, and corn that five leading grain-producing nations produced during 1995 and 1996.

```
   datalines;
1995 BRZ  Wheat    1516
1995 BRZ  Rice     11236
1995 BRZ  Corn     36276
1995 CHN  Wheat    102207
1995 CHN  Rice     185226
1995 CHN  Corn     112331
1995 IND  Wheat    63007
1995 IND  Rice     122372
1995 IND  Corn     9800
1995 INS  Wheat    .
1995 INS  Rice     49860
1995 INS  Corn     8223
1995 USA  Wheat    59494
1995 USA  Rice     7888
1995 USA  Corn     187300
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 CHN  Wheat    109000
1996 CHN  Rice     190100
1996 CHN  Corn     119350
1996 IND  Wheat    62620
```

```
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 INS  Wheat    .
1996 INS  Rice     51165
1996 INS  Corn     8925
1996 USA  Wheat    62099
1996 USA  Rice     7771
1996 USA  Corn     236064
;
```

To view the HTML output and print the PostScript output, close both the HTML and PRINTER destinations. This statement closes the LISTING, HTML and PRINTER destinations and all the files that are associated with them. You must close the HTML destination before you can view the output with a browser. You must close the PRINTER destination before you can print the output on a physical printer. If you do not close these destinations, then output created in subsequent sessions will be routed to them, and you might inadvertently continue to generate both HTML and PostScript output.

```
ods _all_ close;
```

HTML Output

Display 3.2 HTML Frame File Produced by ODS

In this HTML frame file, the object's label, '1996 Grain Production' was supplied by the OBJECTLABEL= suboption. It appears in the table of contents as the link to the output object.

In the body file, the label 'Type of Grain' that was supplied by the LABEL= suboption for the variable TYPE becomes its column heading.

The format for KILOTONS was supplied by the FORMAT= suboption in the FILE statement.

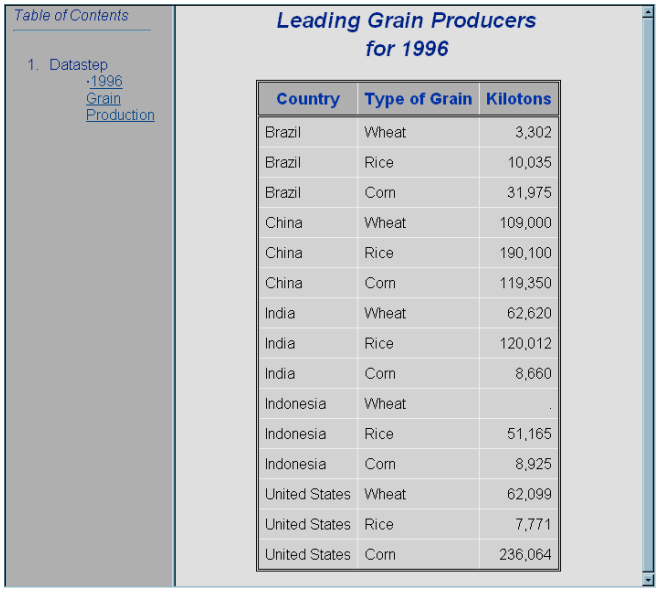


Table of Contents

1. Datastep
   - 1996 Grain Production

Leading Grain Producers for 1996

Country	Type of Grain	Kilotons
Brazil	Wheat	3,302
Brazil	Rice	10,035
Brazil	Corn	31,975
China	Wheat	109,000
China	Rice	190,100
China	Corn	119,350
India	Wheat	62,620
India	Rice	120,012
India	Corn	8,660
Indonesia	Wheat	.
Indonesia	Rice	51,165
Indonesia	Corn	8,925
United States	Wheat	62,099
United States	Rice	7,771
United States	Corn	236,064

Printer Output

Display 3.3 Printer Output Viewed with Ghostview

Just as in the HTML body file and in the listing output, the PostScript output displays the label 'Type of Grain' that was supplied by the LABEL= suboption for the variable TYPE as its column heading.

The format for KILOTONS was supplied by the FORMAT= suboption in the FILE statement.

Leading Grain Producers for 1996

Country	Type of Grain	Kilotons
Brazil	Wheat	3,302
Brazil	Rice	10,035
Brazil	Corn	31,975
China	Wheat	109,000
China	Rice	190,100
China	Corn	119,350
India	Wheat	62,620
India	Rice	120,012
India	Corn	8,660
Indonesia	Wheat	.
Indonesia	Rice	51,165
Indonesia	Corn	8,925
United States	Wheat	62,099
United States	Rice	7,771
United States	Corn	236,064

Listing Output

Just as in the HTML body file and the PostScript output, the listing output displays the label 'Type of Grain' that was supplied by the LABEL= suboption for the variable TYPE. The format for KILOTONS was supplied by the FORMAT= suboption in the FILE statement.

```
                  Leading Grain Producers                    1
                          for 1996
                            Type
                             of
     Country              Grain      Kilotons

  Brazil                  Wheat           3,302
  Brazil                  Rice           10,035
  Brazil                  Corn           31,975
  China                   Wheat         109,000
  China                   Rice          190,100
  China                   Corn          119,350
  India                   Wheat          62,620
  India                   Rice          120,012
  India                   Corn            8,660
  Indonesia               Wheat               .
  Indonesia               Rice           51,165
  Indonesia               Corn            8,925
  United States           Wheat          62,099
  United States           Rice            7,771
  United States           Corn          236,064
```

Example 4: Creating and Using a User-Defined Table Definition Template

ODS features:

- PROC TEMPLATE
- FILE PRINT ODS statement:
  - COLUMNS= suboption:
    - FORMAT= suboption
    - DYNAMIC= suboption
    - GENERIC= suboption
  - TEMPLATE=
- PUT _ODS_ statement:
  - column pointer controls
  - line pointer controls

ODS destination:

- RTF

This example shows how to do the following:

- create a simple user-defined template (table definition) with PROC TEMPLATE
- use a simple user-defined template in the DATA step
- use pointer controls in the PUT _ODS_ statement

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program: Creating the User-Defined Table Definition (Template)

Define the table definition PHONELIST. This PROC TEMPLATE step defines a table definition named PHONELIST.

The template defines two columns: NAME and PHONE.

The GENERIC=ON attribute defines the column for NAME as one that the DATA step can use for multiple variables.

The column definition uses dynamic headers; that is, a variable that uses this column definition takes the value of the header at run time from the DATA step that uses this template. Thus, each variable can have a different column heading.

The STYLE= attribute specifies that the style element DATA be used as the basis for generating the data in this column. The font face and font size that DATA normally uses are replaced by the ones that are specified in the STYLE= attribute.

The header for PHONE is hard-coded as Telephone. The STYLE= attribute specifies a style element to use for the data in this column. For information on PROC TEMPLATE, see Chapter 7, "TEMPLATE Procedure: Overview," on page 395 .

```
proc template;
define table phonelist;
      column name phone;
      dynamic colheader;
   define name;
      generic=on;
      header=colheader;

      style=data{fontstyle=italic fontsize=5};
   end;

   define phone;
      header='Telephone';
      style=datafixed;
   end;
end;
run;
```

Program: Using the User-Defined Template (Table Definition)

Specify that you do not want to produce the default listing output. The ODS LISTING CLOSE statement closes the listing destination to conserve resources. The listing destination is open by default when you open your SAS session.

```
ods listing close;
```

Specify that you want the output formatted in RTF. The ODS RTF statement opens the RTF destination and creates RTF output for use by Microsoft Word. Subsequent output objects are sent to the body file.

```
ods rtf body='your_rtf_file.rtf';
```

Specify a title. The TITLE statement provides a title for the output.

```
title 'New Subscriber Telephone List';
```

Create a format for telephone numbers. PROC FORMAT creates a user-defined format for telephone numbers.

```
proc format;
   picture phonenum .='Not available'
               other='0000)000-0000' (prefix='(');
run;
```

Create the PHONES data set. The data set PHONES contains names and their corresponding phone numbers. Some observations contain missing values for the business or home phone numbers.

```
data phones;
   length first_name $20 last_name $25;
   input first_name $ last_name $ business_phone home_phone;
   datalines;
Jerome Johnson 9193191677 9198462198
Romeo Montague 8008992164 3609736201
Imani Rashid 5088522146 5083669821
Palinor Kent . 9197823199
Ruby Archuleta . .
Takei Ito 7042982145 .
Tom Joad 2099632764 2096684741
;
```

Sort the PHONES data set by last name. PROC SORT sorts the data set PHONES by LAST_NAME and replaces the original data set with the sorted data set.

```
proc sort data=phones;
   by last_name;
run;
```

Begin a DATA step that does not create an output data set. Read an observation from the PHONES data set. Using _NULL_ saves computer resources because it prevents the DATA step from creating an output data set.

```
data _null_;
    set phones;
```

Request that ODS output be created and use the template named PHONELIST. The combination of the fileref PRINT and the ODS option in the FILE statement sends the results of the DATA step to ODS. ODS creates an output object and binds it to the PHONELIST template. Only RTF output is created because only the RTF destination is open.

The TEMPLATE= suboption tells ODS to use the template PHONELIST, which was created previously in the PROC TEMPLATE step.

```
file print ods=(template='phonelist'
```

Place variable values in columns. The COLUMNS= suboption places values of variables into columns that are defined in the template.

Values for both the LAST_NAME and FIRST_NAME variables are written to columns that are defined as NAME in the template.

The GENERIC=ON suboption must be set in both the template and the ODS= option in order for you to use a column definition for more than one column.

The value of the variable BUSINESS_PHONE is placed in a column that is defined as PHONE.

The DYNAMIC= suboption assigns a value to the variable COLHEADER. This value is passed to the template when the output object is created, and the template uses it for the column heading. Thus, even though the variables use the same column definition from the template, the columns in the output object have different column headings.

The FORMAT= suboption assigns the format PHONENUM. to the column named PHONE.

```
columns=
         (name=last_name
           (generic=on
            dynamic=(colheader='Last Name'))
          name=first_name
           (generic=on
            dynamic=(colheader='First Name'))
          phone=business_phone
            (format=phonenum.)
          )
);
```

The following IF/THEN-ELSE statements execute a different PUT _ODS_ statement based on the specified conditions:

- If BUSINESS_PHONE contains missing values, then the PUT statement writes values for LAST_NAME, FIRST_NAME, and BUSINESS_PHONE (the columns that are defined in the ODS= option) into the output buffer. The PUT statement then writes the value for HOME_PHONE in column 3, overwriting the missing value of BUSINESS_PHONE.
- If HOME_PHONE contains a missing value, then the PUT statement simply writes values for LAST_NAME, FIRST_NAME, and BUSINESS_PHONE to the buffer.
- Finally, if both phone numbers have values, then the PUT statement writes values for LAST_NAME, FIRST_NAME, and BUSINESS_PHONE to the buffer in the first line. SAS then goes to the next line (as directed by the line pointer control /) and writes the value of HOME_PHONE in the third column of the next line.

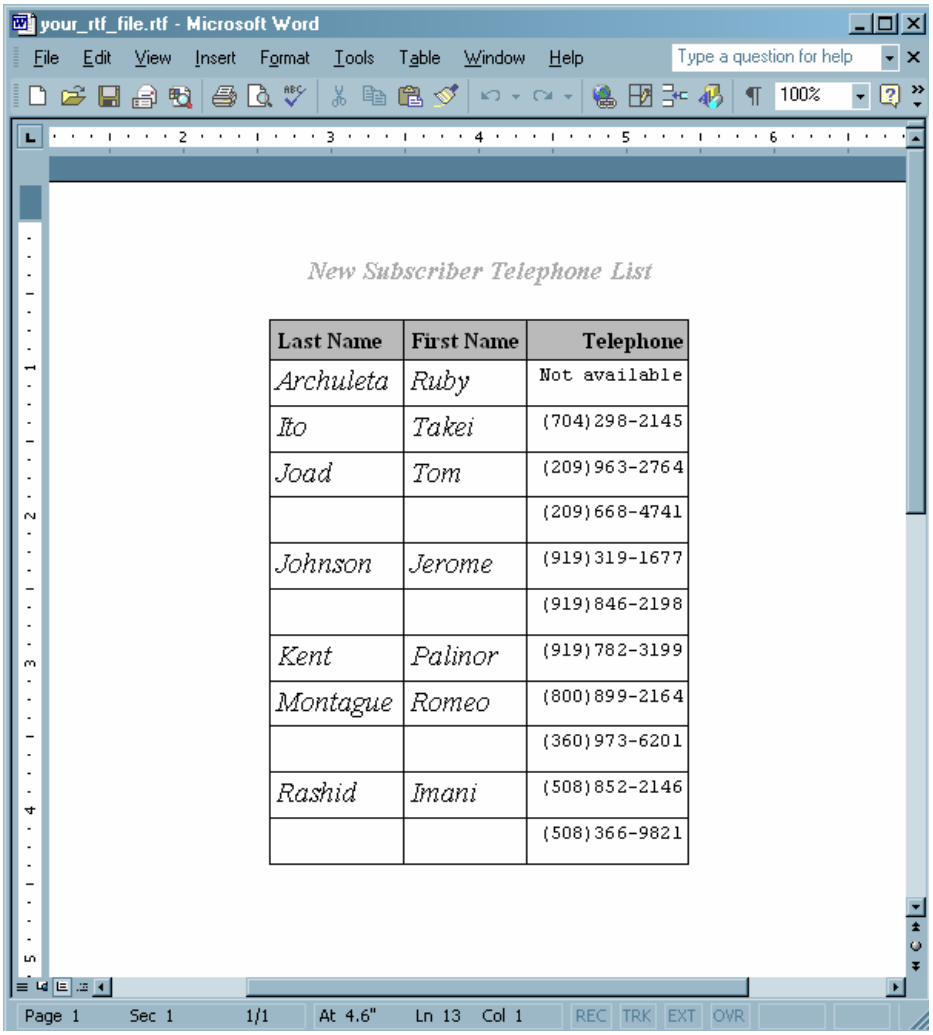
```
if (missing(business_phone)) then
   put _ods_ @3 home_phone;
else if (missing(home_phone)) then
   put _ods_;
else
   put _ods_ / @3 home_phone;
run;
```

Close the RTF destination so that you can view the output. The ODS RTF statement closes the RTF destination and all the files that are associated with it. You must close the destination before you can view the output in Microsoft Word. Also, closing the output prevents all subsequent ODS jobs from automatically producing RTF output.

```
ods RTF close;
```

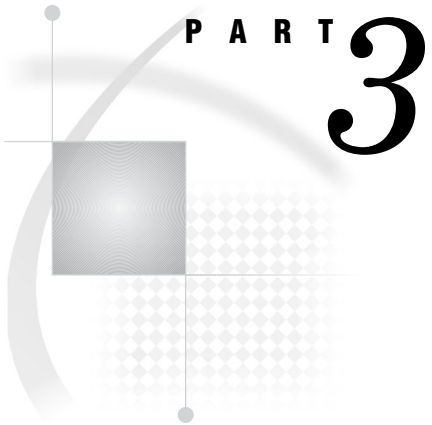
RTF Output

Display 3.4 RTF Output Viewed with Microsoft Word



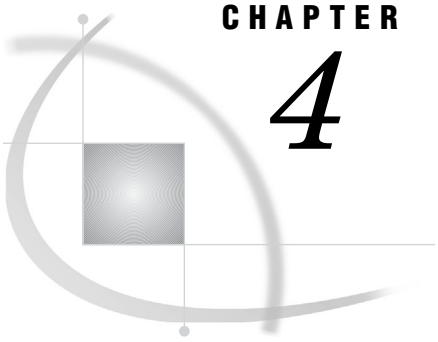
New Subscriber Telephone List

Last Name	First Name	Telephone
Archuleta	*Ruby*	Not available
Ito	*Takei*	(704)298-2145
Joad	*Tom*	(209)963-2764
		(209)668-4741
Johnson	*Jerome*	(919)319-1677
		(919)846-2198
Kent	*Palinor*	(919)782-3199
Montague	*Romeo*	(800)899-2164
		(360)973-6201
Rashid	*Imani*	(508)852-2146
		(508)366-9821



PART 3

ODS Language Statements



CHAPTER

4

Introduction to ODS Language Statements

Definition of ODS Statements

ODS statements provide greater flexibility in generating, storing, and reproducing SAS procedure and DATA step output. You can use the ODS statements to control different features of the Output Delivery System. ODS statements can be used anywhere in your SAS program. Some ODS statements remain in effect until you explicitly change them. Others are automatically cleared at particular times (see the documentation for individual statements).

Types of ODS Statements

DATA Step Statements

DATA step statements are either executable or declarative statements that appear in the DATA step. The ODS statements that are used in the DATA step are executable statements. Executable statements result in some action during individual iterations of the DATA step. For information about declarative statements, see *SAS Language Reference: Dictionary*.

Global Statements

Global statements

- provide information to SAS
- request information or data
- move between different modes of execution
- set values for system options

The global ODS statements deliver or store output in a variety of formats. You can use global statements anywhere in a SAS program. Global statements are not executable; they take effect as soon as SAS compiles program statements.

Global ODS statements are organized into three categories:

ODS: Output Control
statements that provide descriptive information about the specified output objects and indicate whether the style definition or table definition is supplied by SAS. The Output Control statements can do the following:

- select or exclude specific output objects for specific destinations
- specify the location where you want to search for or store style definitions or table definitions
- verify if you are using a style definition or a table definition that is supplied by SAS
- provide descriptive information about each specified output object, such as name, label, template, path, and label path

ODS: SAS Formatted
statements that enable you to produce items that are specifuc to SAS, such as a SAS data set, SAS output listing, or an ODS document. The statements in the ODS SAS Formatted category create the SAS entities. For more information, see "The SAS Formatted Destinations" on page 25.

ODS: Third-Party Formatted
statements that enable you to apply styles and markup languages, or produce output to physical printers using page description languages. For more information, see "The Third-Party Formatted Destinations" on page 26.

Procedure Statements

For information about the TEMPLATE procedure, see Chapter 7, "TEMPLATE Procedure: Overview," on page 395. For information about the DOCUMENT procedure, see Chapter 6, "The DOCUMENT Procedure," on page 333.

ODS Statement Category Descriptions

The following table lists and describes the categories of ODS global statements:

Table 4.1 Global Statements by Category

Statement Category	Function
ODS: Output Control	Provide descriptive information about the specified output objects and their locations.
ODS: SAS Formatted	Produce listing output, a SAS output data set, or a hierarchy file.
ODS: Third-Party Formatted	Produce files that are formatted in the proper destination format.

ODS Statements by Category

Table 4.2 Categories and Descriptions of ODS Statements

Category	Dictionary of ODS Language Statements	Description
Data Access	"ODS PACKAGE Statement" on page 198	The ODS PACKAGE statement opens, adds to, publishes, or closes one SAS Output Delivery System (ODS) package object.
File-handling	"FILE Statement for ODS" on page 68	Creates an ODS output object by binding the data component to the table definition (template). As an option, the FILE Statement lists the variables to include in the ODS output, and it specifies options that control the way that the variables are formatted.
	"PUT Statement for ODS" on page 81	Writes data values to a special buffer from which they can be written to the data component and then formatted by ODS.
ODS: Output Control	"LIBNAME Statement, SASEDOC" on page 77	Uses the SASEDOC engine to associate a SAS libref (library reference) with one or more ODS output objects that are stored in an ODS document.
	"ODS _ALL_ CLOSE Statement" on page 85	Closes all open ODS output destinations.
	"ODS DOCUMENT Statement" on page 94	Opens, manages, or closes the DOCUMENT destination, which produces a hierarchy of output objects that enables you to produce multiple ODS output formats without rerunning a PROC or DATA step.
	"ODS ESCAPECHAR Statement" on page 97	Defines a representative character to be used in output strings.
	"ODS EXCLUDE Statement" on page 110	Specifies output objects to exclude from ODS destinations.
		Enables or disables ODS graphics processing and sets graphics environment options. This statement affects ODS template-based graphics only. The ODS GRAPHICS statement does not affect device-based graphics.
	"ODS PATH Statement" on page 206	Specifies locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them.
	"ODS PROCLABEL Statement" on page 237	Enables you to change a procedure label.
	"ODS PROCTITLE Statement" on page 238	Determines whether to write the title that identifies the procedure that produces the results in the output.
	"ODS RESULTS Statement" on page 241	Tracks ODS output in the Results window.
	"ODS SELECT Statement" on page 264	Specifies output objects for ODS destinations.

Category	Dictionary of ODS Language Statements	Description
	"ODS SHOW Statement" on page 277	Writes the specified selection or exclusion list to the SAS log.
	"ODS TEXT= Statement" on page 313	Inserts text into your ODS output.
	"ODS TRACE Statement" on page 317	Writes to the SAS log a record of each output object that is created, or suppresses the writing of this record.
	"ODS USEGOPT Statement" on page 322	Determines whether ODS uses traditional SAS/GRAPH option settings.
	"ODS VERIFY Statement" on page 325	Prints or suppresses a message indicating that a style definition or a table definition being used is not supplied by SAS.
ODS: SAS Formatted	"ODS DECIMAL_ALIGN Statement" on page 91	Controls the justification of numeric columns when no justification is specified.
	"ODS LISTING Statement" on page 143	Opens, manages, or closes the LISTING destination.
	"ODS OUTPUT Statement" on page 184	Produces a SAS data set from an output object and manages the selection and exclusion lists for the OUTPUT destination.
ODS: Third-Party Formatted	"ODS CHTML Statement" on page 85	Opens, manages, or closes the CHTML destination, which produces a compact, minimal HTML that does not use style information.
	"ODS CSVALL Statement" on page 88	Opens, manages, or closes the CSVALL destination, which produces HTML output containing columns of data values that are separated by commas, and produces tabular output with titles, notes, and bylines.
	"ODS DOCBOOK Statement" on page 91	Opens, manages, or closes the DOCBOOK destination, which produces XML output that conforms to the DocBook DTD by OASIS.
	"ODS HTML Statement" on page 124	Opens, manages, or closes the HTML destination, which produces HTML 4.0 output that contains embedded style sheets.
	"ODS HTMLCSS Statement" on page 135	Opens, manages, or closes the HTMLCSS destination, which produces HTML output with cascading style sheets.
	"ODS HTML3 Statement" on page 137	Opens, manages, or closes the HTML3 destination, which produces HTML 3.2 formatted output.
	"ODS IMODE Statement" on page 140	Opens, manages, or closes the IMODE destination, which produces HTML output as a column of output, separated by lines.
	"ODS MARKUP Statement" on page 147	Opens, manages, or closes the MARKUP destination, which produces SAS output that is formatted using one of many different markup languages.
	"ODS PCL Statement" on page 208	Opens, manages, or closes the PCL destination, which produces printable output for PCL (HP LaserJet) files.

Category	Dictionary of ODS Language Statements	Description
	"ODS PDF Statement" on page 210	Opens, manages, or closes the PDF destination, which produces PDF output, a form of output that is read by Adobe Acrobat and other applications.
	"ODS PHTML Statement" on page 215	Opens, manages, or closes the PHTML destination, which produces simple HTML output that uses twelve style elements and no class attributes for the presentation. Class attributes are used only for the justification.
	"ODS PRINTER Statement" on page 218	Opens, manages, or closes the PRINTER destination, which produces printable output.
	"ODS PS Statement" on page 239	Opens, manages, or closes the PS destination, which produces PostScript (PS) output.
	"ODS RTF Statement" on page 242	Opens, manages, or closes the RTF destination, which produces output written in Rich Text Format for use with Microsoft Word 2002.
	"ODS Tagset Statement" on page 278	Opens, manages, or closes the specified tagset destination.
	"ODS TAGSETS.RTF Statement" on page 286	Opens, manages, or closes the RTF destination, which produces measured output that is written in Rich Text Format for use with Microsoft Word 2002.
	"ODS WML Statement" on page 326	Opens, manages, or closes the WML destination, which uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a simple list for a table of contents.

CHAPTER

5

Dictionary of ODS Language Statements

FILE Statement for ODS

Creates an ODS output object by binding the data component to the table definition (template). As an option, the FILE Statement lists the variables to include in the ODS output, and it specifies options that control the way that the variables are formatted.

Valid: in a DATA step

Category: File-handling

Type: Executable

Default: ODS sends the output object to all open ODS destinations.

Syntax

FILE PRINT ODS <=(*ODS-suboption(s)*)><*options*> ;

Note: This syntax shows only the ODS form of the FILE statement. For the complete syntax, see the FILE statement in *SAS Language Reference: Dictionary*. △

Required Arguments

PRINT
is a reserved fileref that you must use when you direct output to ODS.

Requirement: You must use PRINT in a FILE statement that uses the ODS option.

Featured in: "Example 1: Creating a Report with the DATA Step and the Default Table Definition" on page 41

ODS<=(*ODS-suboptions*)>
defines the structure of the data component and binds the data component to a table definition. The result is an ODS output object. ODS sends this object to all open ODS destinations.

See also: "ODS Suboptions" on page 69 for information about the ODS suboptions

Featured in: All examples

Options

N=*number*
specifies the number of lines that are available to the output pointer in the current iteration of the DATA step.

overflow-control
determines the PUT statement behavior when the output pointer attempts to move past the last ODS column in the buffer.

overflow-control is one of the following:

DROPOVER
discards items when a PUT statement attempts to write beyond the last ODS column in the buffer. A message in the log at the end of the DATA step informs you if data was not written to the buffer.

FLOWOVER
moves the output pointer to a new line if a PUT statement attempts to write an item beyond the last ODS column in the buffer. The PUT statement writes the next item in the first ODS column of the new line.

STOPOVER
stops processing the DATA step immediately if a PUT statement attempts to write beyond the last ODS column in the buffer. SAS discards the data item, writes the portion of the buffer that was built before the error occurred, and issues an error message.

Default: FLOWOVER

Without ODS Suboptions

If you do not specify any ODS suboptions, the DATA step uses a default table definition (BASE.DATASTEP.TABLE) that is stored in the SASHELP.TMPLMST template store. This definition defines two generic columns: one for character variables and one for numeric variables. ODS associates each variable in the DATA step with one of these columns and displays the variables in the order in which they are defined in the DATA step.

If there are no suboptions, the default table definition uses the variable's label as its column heading. If no label exists, the definition uses the variable's name as the column heading.

ODS Suboptions

Task	Suboption
Specify one or more columns for the data component	COLUMNS= or VARIABLES=
Specify default values for dynamic-attribute values	DYNAMIC=
Specify whether all column definitions in the table definition can be used by more than one variable	GENERIC=
Specify a column heading to use for any column that does not have a column heading specified in the COLUMNS= or VARIABLES= suboption	LABEL=
Specify a name for the output object that the DATA step produces	OBJECT=

Task	Suboption
Specify a label for the output object that the DATA step produces	OBJECTLABEL=
Specify the table definition to use with the data component to produce the output object	TEMPLATE=

COLUMNS=(*column-specification(s)*)
specifies one or more columns for the data component and determines their order in the data component.

Restriction: You can use only one COLUMNS= suboption in a FILE PRINT ODS statement.

Restriction: You can use either the COLUMNS= suboption or the VARIABLES= suboption, but not both, in a single FILE PRINT ODS statement.

Requirement: You must enclose a *column-specification* in parentheses.

Tip: The order of the columns in the output object is determined by their order in the table definition, not by their order in the data component.

Tip: To override the default order, use the ORDER_DATA= table attribute in the PROC TEMPLATE step that creates the definition. The default DATA step table definition uses this attribute. For more information, see the discussion of ORDER_DATA= table attribute.

Tip: If you do not specify COLUMNS= or VARIABLES=, then the order of columns in the data component matches the order of the corresponding variables in the program data vector.

Each *column-specification* associates a DATA step variable with a column that is defined in the table definition. *column-specification* has this general form:

(*column-name-1*<=*variable-name-1*<(*attribute-suboptions*)>> <…
column-name-n<=*variable-name-n*<(*attribute-suboptions*)>>>)

column-name
is the name of a column. This name must match the name that is defined in the table definition that you use.

Restriction: *column-name* must conform to the rules for SAS variable names. For information, see the *SAS Language Reference: Dictionary*.

Requirement: You must enclose a *column-name* in parentheses.

Tip: You can use list notation (for example, **score1-score5**) to specify multiple column names.

Featured in: “Example 4: Creating and Using a User-Defined Table Definition Template” on page 53

variable-name
specifies a variable in the DATA step to place in the specified column.

Default: If you omit *variable-name*, then ODS looks for a DATA step variable named *column-name* to place in the specified column. If no such variable exists, then ODS returns an error.

Tip: You can use list notation (for example, **score1-score5**) to specify a range of variable names.

Featured in: “Example 4: Creating and Using a User-Defined Table Definition Template” on page 53

(*attribute-suboptions*)
assigns a characteristic, such as a label or a format, to a particular column in the data component. These individual specifications override any attributes that are set by the DATA step.

The following table lists the attribute suboptions that are available for the COLUMNS= suboption. For a complete description, see "Attribute Suboptions" on page 75.

Task	Attribute Suboption
Specify a value for the variable defined by the DYNAMIC statement in a table template	DYNAMIC=
Specify a format for the current column	FORMAT=
Specify whether the DATA step uses this column definition for multiple variables	GENERIC=
Specify a label for a particular column	LABEL=

Requirement: You must enclose *attribute-suboptions* in parentheses.

DYNAMIC=(*dynamic-specification(s)*)
specifies default values for dynamic-attribute values.

A dynamic-attribute value is defined in the table definition. Its name serves as a placeholder for the value that is supplied to the data component with the DYNAMIC= suboption. When ODS creates the output object from the table definition and the data component, it substitutes the appropriate value from the data component for the value's name in the table definition.

Each *dynamic-specification* has the following form:

dynamic-value-name<=*variable-name* | *constant*>

dynamic-value-name
is the name that the table definition gives to a dynamic-attribute value.

variable-name
specifies a variable whose value is assigned to *dynamic-value-name* and passed to ODS to substitute for the placeholder in the table definition when it creates the output object.

constant
specifies a constant to assign to *dynamic-value-name* and pass to ODS to substitute for the placeholder in the table definition when it creates the output object.

Default: By default, the DYNAMIC= suboption applies to all columns in the data component.

Interaction: Columns that do not contain their own DYNAMIC= suboption specifications use these *dynamic-specifications*.

Tip: You can override the default specification for an individual column by specifying the DYNAMIC= suboption as an attribute for that column in the COLUMNS= or the VARIABLES= suboption.

See also: "DYNAMIC Statement" on page 619

GENERIC=ON | OFF
indicates whether the DATA step uses all column definitions for multiple variables.

ON

indicates that the DATA step uses all column definitions for multiple variables.

OFF
indicates that the DATA step uses no column definitions for multiple variables.

Default: OFF

Default: By default, the GENERIC= suboption applies to all columns in the data component.

Restriction: ODS does not recognize the column names as a match unless you specify the (COLUMNS=(GENERIC=ON)) suboption.

Interaction: If you do not specify a table definition, the GENERIC= suboption is set to ON.

Tip: To override the default specification for an individual column, specify the GENERIC= suboption as an attribute for that column in the COLUMNS= or the VARIABLES= suboption.

Tip: The GENERIC= option in the DATA step is used in conjunction with the GENERIC= column attribute in the table template. See the GENERIC= column attribute in "Column Attributes" on page 601.

LABEL='*column-label*'
specifies a label for any column that does not have a label specified in the COLUMNS= or VARIABLES= suboption.

Default: If you use the LABEL= suboption, ODS uses the first of these labels that it finds:

- a label that is specified with HEADER= attribute for a particular column in the table definition (see HEADER= on page 607 column attribute)
- a label that is specified for a particular column with LABEL= suboption in the COLUMNS= or VARIABLES= suboption
- a label that is specified with LABEL= suboption in the ODS= option
- a label that is assigned with the LABEL statement in the DATA step

Tip: If you omit the LABEL= suboption, the contents of the table definition determines whether the column heading contains the variable name or is blank.

Featured in: "Example 3: Assigning Attributes to Columns in ODS Output" on page 48

OBJECT= *object-name*
specifies a name for the output object.

The Results window and the HTML contents file both contain a description of, and a link to, each output object. The description contains the first of the following items that ODS finds:

- the object's label
- the current title if it is not the default title, "The SAS System"
- the object's name
- the string **`FilePrint#`**, in which # increases by 1 for each DATA step that you run in the current SAS process without specifying an object name or an object label

Restriction: *object-name* must conform to the rules for SAS variable names. For information about these rules, see Rules for Words and Names in the SAS Language in *SAS Language Reference: Concepts*.

OBJECTLABEL='*object-label*'
specifies a label for the output object.

The Results window and the HTML contents file both contain a description of, and a link to, each output object. The description contains the first of the following items that ODS finds:

- the object's label
- the current title if it is not the default title, "The SAS System"
- the object's name (see OBJECT= on page 72)
- the string **`FilePrint#`**, in which # increases by 1 for each DATA step that you run in the current SAS process without specifying an object name or an object label

Requirement: You must enclose an *object-label* in quotation marks.

Featured in: "Example 3: Assigning Attributes to Columns in ODS Output" on page 48

TEMPLATE= '*table-definition-name*'
specifies the table definition to use with the data component to produce the output object.

table-definition-name
is the path to the table definition. SAS stores a table definition as an item in an item store.

Default: If you do not specify the TEMPLATE= option, ODS uses BASE.DATASTEP.TABLE, the default table definition.

Default: If you do specify the TEMPLATE= suboption, ODS first looks for *table-definition-name* in SASUSER.TEMPLAT, and then it looks in SASHELP.TMPLMST.

Requirement: You must enclose a *table-definition-name* in quotation marks.

Interaction: When you use the default table definition, the GENERIC= suboption is set to ON for all columns in the data component. For more information, see GENERIC= on page 71.

Tip: When you use the BASE.DATASTEP.TABLE template, character values are left-justified. If you want character values to be right-justified, specify the BASE.DATASTEP.TABLENOJUST template.

Tip: You can change the locations in which ODS searches for the *table-definition-name* by using the "ODS PATH Statement" on page 206.

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

VARIABLES=(*variable-specification(s)*)
specifies one or more columns for the data component of the output object. Each *variable-specification* associates a DATA step variable with a column that is defined in the table definition. The *variable-specification* value has this general form:

(*variable-name-1*<=*column-name-1*<(*attribute-suboptions*)>> <…
variable-name-n<=*column-name-n*<(*attribute-suboptions*)>>>)

variable-name
specifies a variable in the DATA step to use as a column in the data component.

Tip: You can use list notation (for example, **`score1-score5`**) to specify a range of variable names.

Featured in: "Example 2: Producing ODS Output That Contains Selected Variables" on page 44 and "Example 3: Assigning Attributes to Columns in ODS Output" on page 48

column-name
is the name of a column. This name must match a name that is defined in the table definition.

Default: If you are using the default table definition and you omit *column-name*, then ODS uses the variable label to name the column. If the variable has no label, then ODS uses the variable name.

Default: If you use a table definition other than the default table definition and you omit *column-name*, ODS looks in the table definition for a column that is named *variable-name* and places the variable in that column. ODS returns an error if no such column exists.

Restriction: *column-name* must match a column name in the table definition that you are using. It must also conform to the rules for SAS variable names. For information about these rules, see Rules for Words and Names in the SAS Language in *SAS Language Reference: Concepts*.

Tip: You can use list notation (for example, **`score1-score5`**) to specify a range of column names.

(*attribute-suboptions*)
assigns a characteristic, such as a label or a format, to a particular column in the data component. These individual specifications override any attributes that are set in the DATA step for the entire data component.

The following table lists the attribute suboptions available for the VARIABLES= suboption. For a complete description, see "Attribute Suboptions" on page 75.

Task	Attribute Suboption
Specify a value for the variable defined by the DYNAMIC statement in a table template	DYNAMIC=
Specify a format for the current column	FORMAT=
Specify whether the DATA step uses this column definition for multiple variables	GENERIC=
Specify a label for a particular column	LABEL=

Default: If you specify the VARIABLES= suboption, the order of the columns in the output object is determined by their order in the table definition, not by their order in the data component. If you do not specify COLUMNS= or VARIABLES= suboptions, the order of columns in the data component matches the order of the corresponding variables in the program data vector.

Restriction: You can use only one VARIABLES= suboption in a FILE PRINT ODS statement.

Restriction: You can use either the COLUMNS= suboption or the VARIABLES= suboption to associate variables with columns, but you cannot use both suboptions in the same FILE PRINT ODS statement.

Tip: To override the default order, use the ORDER_DATA table attribute in the PROC TEMPLATE step that creates the definition. The default DATA step table definition uses this attribute. For more information see the ORDER_DATA= table attribute.

Tip: The VARIABLES= suboption is for use primarily with the default DATA step table definition. When you use the default definition, the DATA step can map variables to the appropriate column in the definition so you do not need to specify a column name.

Featured in: "Example 2: Producing ODS Output That Contains Selected Variables" on page 44 and "Example 3: Assigning Attributes to Columns in ODS Output" on page 48.

Attribute Suboptions

DYNAMIC=*dynamic-specification(s)*
specifies a value for the variable defined by the DYNAMIC statement in a table template.

Main discussion: DYNAMIC= on page 71

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

See also: "DYNAMIC Statement" on page 619

FORMAT=*format-name*
specifies a format for the current column.

Default: ODS uses the first of these formats for the variable that it finds:

- for nongeneric columns, a format that is specified in the column definition
- a format that is specified in the FORMAT= column attribute
- a format that is specified in a FORMAT statement
- the default format (w. for character variables; BEST12. for numeric variables)

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

Note: Formats for generic columns that are specified in the table definition are ignored by the DATA step interface to ODS. △

GENERIC=ON | OFF
specifies whether the DATA step uses this column definition for multiple variables.

Default: OFF

Main discussion: GENERIC= on page 71

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

See also: GENERIC on page 607

Tip: The GENERIC= option in the DATA step is used in conjunction with the GENERIC= column attribute in the table template. See the GENERIC= column attribute in "Column Attributes" on page 601.

LABEL='*column-label*'
specifies a label for the specified column.

Main discussion: LABEL= on page 72

Featured in: "Example 3: Assigning Attributes to Columns in ODS Output" on page 48

Details

Restrictions When Using the FILE Statement with ODS

The following restrictions apply to the FILE statement when you use it with ODS:

- These arguments affect only listing output:
  - FOOTNOTES and NOFOOTNOTES
  - LINESIZE
  - PAGESIZE
  - TITLE and NOTITLES
- Do not use these arguments:
  - DELIMITER=
  - DLMSTR=
  - DSD
  - _FILE_=
  - FILEVAR=
  - HEADER=
  - PAD

Using Options and Suboptions Options apply to all columns and suboptions apply to specific columns.

For example, both of the following DATA steps produce the same output. This DATA step specifies the suboption GENERIC=ON for every column.

Example Code 5.1 Data Step Using the GENERIC=ON Suboption

```
data _null_;
   set top3list;
      file print ods = (
      template='means.topn'
      columns=(
         class=school(generic=on)
         class=year(generic=on)
         sum=moneyRaised_sum(generic=on)
         mean=moneyRaised_mean(generic=on)
         raised=moneyRaised_1(generic=on)
         raised=moneyRaised_2(generic=on)
         raised=moneyRaised_3(generic=on)
         name=name_1(generic=on)
         name=name_2(generic=on)
         name=name_3(generic=on)
         school=school_1(generic=on)
         school=school_2(generic=on)
         school=school_3(generic=on)
         year=year_1(generic=on)
         year=year_2(generic=on)
         year=year_3(generic=on)
         )
      );

    put _ods_;
run;
```

This DATA step uses the GENERIC=ON option, which has to be specified only once.

Example Code 5.2 Data Step Using the GENERIC=ON Option

```
data _null_;
   set top3list;
      file print ods = (
      template='means.topn'
      generic=on
      columns=(
         class=school
         class=year
         sum=moneyRaised_sum
         mean=moneyRaised_mean
         raised=moneyRaised_1
         raised=moneyRaised_2
         raised=moneyRaised_3
         name=name_1
         name=name_2
         name=name_3
         school=school_1
         school=school_2
         school=school_3
         year=year_1
         year=year_2
         year=year_3
         )
      );

    put _ods_;
run;
```

See Also

Statement:

"PUT Statement for ODS" on page 81

Chapter 3, "Output Delivery System and the DATA Step," on page 39

"Examples" on page 41

LIBNAME Statement, SASEDOC

Uses the SASEDOC engine to associate a SAS libref (library reference) with one or more ODS output objects that are stored in an ODS document.

Valid: anywhere

Category: ODS: Output Control

Restriction: The LIBNAME statement that is used with the SASEDOC engine provides read access to an output object. You cannot write an output object to a library with the SASEDOC engine, but you can delete or rename a data set.

Syntax

LIBNAME *libref* **SASEDOC** '*path*' <sasedoc-engine-option> <*options*>;

Required Arguments

libref
is a shortcut name or a nickname for the aggregate storage location where your SAS files are stored. It is any SAS name that you choose for assigning a new libref. When you are disassociating a libref from a SAS library, or when you are listing attributes, specify a libref that was previously assigned or else use the CLEAR argument.

Tip: The association between a libref and a SAS library lasts only for the duration of the SAS session or until you change it or discontinue it with another LIBNAME statement for the same libref.

SASEDOC
is the name of the engine that associates a SAS libref (library reference) with one or more ODS output objects that are stored in an ODS document.

path
is the fully specified location of an ODS document directory.

SASEDOC Engine Options

DOC_SEQNO=*sequence-number*
permits you to specify the sequence number of the output object to be accessed. This is necessary when multiple output objects that are in the same directory have the same name. By default, the SASEDOC LIBNAME engine can access only the most recently created output object, which might not be the one that you want to access. Specify DOC_SEQNO to override the default.

sequence-number
is a number which, when combined with a pathname, uniquely identifies the entry in the directory.

See also: "Understanding Sequence Numbers" on page 366

Additional LIBNAME Statement Arguments and Options

For additional arguments and options that are valid for the LIBNAME statement, see the LIBNAME statement in *SAS Language Reference: Dictionary*.

Details

Using the LIBNAME Statement The SASEDOC LIBNAME engine permits you to access output objects that are stored in an ODS document. A data set that is accessed by using the SASEDOC LIBNAME engine might differ structurally from one created by replaying the ODS document output object to the ODS OUTPUT destination. This is because the ODS OUTPUT destination recognizes the output object's template, but the SASEDOC LIBNAME engine does not.

Examples

Example 1: Assigning a LIBNAME to an ODS DOCUMENT

LIBNAME statement:

Option:

DOC_SEQNO=

ODS DOCUMENT statement:

Option:

NAME=

Other SAS features:

PROC DATASETS

PROC GLM

PROC PRINT

Data Sets:

"Creating the Plants Data Set" on page 880.

"Creating the Plant_Stat Data Set" on page 880.

Program Description This example assigns a libref to an ODS document directory that contains four output objects created by PROC GLM. The four output objects are tables:

Overall ANOVA

Fit statistics

Type I model ANOVA

Type III model ANOVA

Program

Create the ODS document *sasuser.odsglm* and open the DOCUMENT destination. The ODS DOCUMENT statement opens the DOCUMENT destination. The NAME= option assigns the name **sasuser.odsglm** to the ODS document that will contain the output from the PROC GLM program. The access-option WRITE provides Write access to the document. Note that **odsglm** will be created in the SASUSER library.

```
ods document name=sasuser.odsglm(write);
```

Create the output objects. The GLM procedure creates the output objects. The Plant_Stats data set contains the statistical information that PROC GLM uses to create the output objects. For information about viewing a record of each output object that is created, see the "ODS TRACE Statement" on page 317.

```
proc glm  data=plant_stats;
  class month;
  model age age2 age3=month / nouni;
  manova h=month /print;
run;
```

Create the output objects. The GLM procedure creates the output objects. The Plants data set contains the statistical information that PROC GLM uses to create the output objects. For information about viewing a record of each output object that is created, see "ODS TRACE Statement" on page 317.

```
proc glm  data=plants order=data;
   class type block;
   model stemleng=type block;
   means type;
   contrast 'compost vs others'  type -1 -1 -1 -1  6 -1 -1;
   contrast 'river soils vs.non' type -1 -1 -1 -1  0  5 -1,
                                 type -1  4 -1 -1  0  0 -1;
   contrast 'glacial vs drift'   type -1  0  1  1  0  0 -1;
   contrast 'clarion vs webster' type -1  0  0  0  0  0  1;
   contrast 'knox vs oneill'     type  0  0  1 -1  0  0  0;
quit;
```

Close the DOCUMENT destination. If you do not close the DOCUMENT destination, you will be unable to see DOCUMENT procedure output.

```
ods document close;
```

Associate the libref *mylib* with the directory *stemleng*. The LIBNAME statement uses the SASEDOC engine to associate the SAS libref **mylib** with the directory **stemleng** that is stored in the ODS document **sasuser.odsglm**. Notice that the path includes **anova#1** and not just **anova**. This is because there are two **anova** directories, and this code is specifying the first directory. If the sequence number was omitted, then ODS would associate the libref with the second directory.

```
libname mylib sasedoc '\sasuser.odsglm\glm\anova#1\stemleng';
```

The LIBRARY= option specifies **mylib** as the procedure input library. The QUIT statement stops the DATASETS procedure.

```
proc datasets lib=mylib;
run;
quit;
```

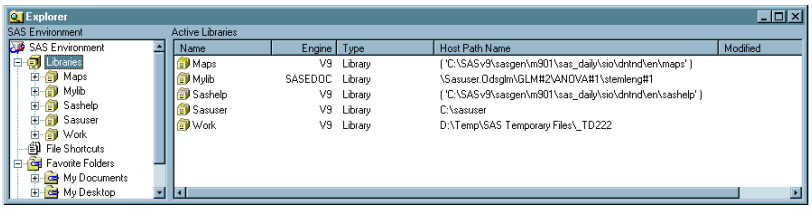
Print the data sets. Since two output objects have the same name (ModelANOVA), the SASEDOC LIBNAME engine recognizes only the second table, because it was created more recently than the first table. The DOC_SEQNO= data set option specifies a sequence number of 1 in order to access the first table .

```
proc print data=mylib.modelanova;
run;
proc print data=mylib.modelanova(doc_seqno=1);
run;
```

Output

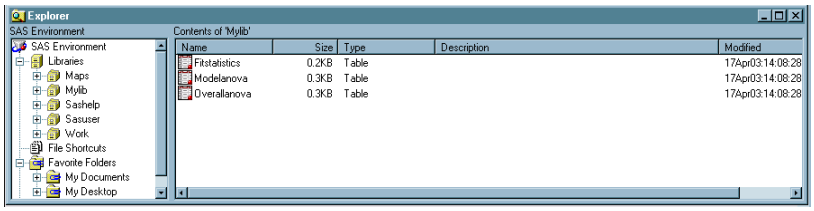
Display 5.1 Explorer Window

The following display shows the Explorer window that contains the SAS library **Mylib** which is associated with the directory **stemleng**. The **stemleng** directory is stored in the ODS document **sasuser.odsglm**.



Display 5.2 The Contents of Mylib

The following display shows the Explorer window that contains the contents of the SAS library **Mylib**. The three output objects are actually stored in an ODS document.



See Also

Procedures:

Chapter 6, “The DOCUMENT Procedure,” on page 333

Statements:

“ODS DOCUMENT Statement” on page 94

“ODS TRACE Statement” on page 317

PUT Statement for ODS

Writes data values to a special buffer from which they can be written to the data component and then formatted by ODS.

Valid: in a DATA step

Category: File-handling

Type: Executable

Requirement: If you use the _ODS_ option in the PUT statement, then you must use the FILE PRINT ODS statement.

Syntax

PUT <*specification*><_ODS_><@|@@>;

Note: This syntax shows only the ODS form of the PUT statement when you are binding to a template. For the complete syntax, see the PUT statement in *SAS Language Reference: Dictionary*. △

Options

specification
specifies one or more variables to write and where to write them. Specification has the following form:

<*ods-pointer-control-1*> *variable-1* <...<*ods-pointer-control-n*>*variable-n*>

ods-pointer-control
moves the pointer in the buffer to a specified line or column.

See also: "When the Pointer Moves Past the End of a Line" on page 84

variable
identifies the variable to write.

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

ODS
specifies that the PUT statement writes values to the data component for each of the variables that were defined as columns with the FILE PRINT ODS COLUMNS= statement.

Default: The order of these columns is determined by the order that is specified by the COLUMNS= suboption in the FILE PRINT ODS statement. If you omit the COLUMNS= suboption, then the order of the variables in the program data vector determines their order in the output object.

Requirement: If you specify the _ODS_ option, then you must use the FILE PRINT ODS statement and the FILE PRINT ODS statement must precede the PUT _ODS_ statement. For more information, see ODS<=(ODS-suboptions)> on page 68.

Interaction: You can use _ODS_ in a PUT statement that specifies the placement of individual variables. _ODS_ writes to a particular row and column only if another PUT statement has not already written a variable to that same row and column. The position of _ODS_ in the PUT statement does not affect the outcome in the data component.

Tip: By default, the order of the columns in the data component matches the order of the columns in the buffer. However, if you have specified a table definition, it might override this order. For more information, see the discussion of ORDER_DATA in Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593 .

@ | @@
holds an output line for the execution of the next PUT statement across iterations of the DATA step. The line-hold specifiers are called *trailing* @ and *double trailing* @.

Default: If you do not use @ or @@, then each PUT statement in a DATA step writes a new line to the buffer.

Main discussion: "When the Pointer Moves Past the End of a Line" on page 84

Details

ODS Column Pointer Controls ODS column pointer controls differ slightly from column pointer controls in a PUT statement that does not use ODS. An ODS column refers not to a single character space but to a column that contains an entire variable value. Therefore, an ODS column pointer control moves from one entire value to the next, not from one character space to another. Column 1 contains values for the first variable in the output; column 2 contains values for the second variable, and so on.

ODS column pointer controls have the following general forms:

@ods-column
moves the pointer to the specified ODS column. *ods-column* is a number, a numeric variable, or an expression that identifies the column to write to.

Requirement: If *ods-column* is a number, then it must be a positive integer.

If *ods-column* is a numeric variable or an expression, then SAS treats it as follows:

Variable or Expression	SAS Response
Not an integer	Truncates the decimal portion and uses only the integer value.
0 or negative	Moves the pointer to column 1.

Default: If *ods-column* exceeds the number of columns in the data component, then ODS does the following:

1. writes the current line
2. moves the pointer to the first ODS column on the next line
3. continues to process the PUT statement

Tip: You can alter the default behavior with options in the FILE PRINT ODS statement. For more information, see the discussion of overflow control on page 68.

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

+ods-column
moves the pointer by the specified number of ODS columns. *ods-column* is a number, a numeric variable, or an expression that specifies the number of columns to move the pointer.

Requirement: If *ods-column* is a number, then it must be an integer.

If *ods-column* is a numeric variable or an expression, then it does not have to be an integer. If it is not an integer, then SAS truncates the decimal portion and uses only the integer value.

ods-column	SAS Response
A positive integer	Moves the pointer to the right.
A negative integer	Moves the pointer to the left.
0	Pointer does not move.

Tip: If the current column position becomes less than 1, then the pointer moves to column 1. If the current column position exceeds the number of columns in the data component, then ODS does the following:

1. writes the current line
2. moves the pointer to the first ODS column on the next line
3. continues to process the PUT statement

See also: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

@ '*column-name*'
moves the pointer to the ODS column identified by '*column-name*'. The column name is a data component variable name.

Requirement: *column-name* must be enclosed in quotation marks.

ODS Line Pointer Controls Line pointer controls in a DATA step that uses ODS are the same as line pointer controls in a DATA step that does not use ODS. However, you can use only those listed below with ODS. Line pointer controls have the following general forms:

#*line*
moves the pointer to the specified line. *line* is a number, a numeric variable, or an expression that identifies the line that specifies where to write.

Requirement: If *line* is a number, then it must be an integer. If *line* is a numeric variable or an expression, it does not have to be an integer. If it is not an integer, then SAS truncates the decimal portion and uses only the integer value.

/
moves the pointer to the first column of the next line.

Featured in: "Example 4: Creating and Using a User-Defined Table Definition Template" on page 53

Note: If you use a line pointer control to skip lines in ODS output, then SAS sets to a missing value all columns that are not referenced on the current line or skipped lines to a missing value. Columns that contain numeric values will display a period for the missing value. If you prefer not to include these periods in your ODS output, you can display missing numeric values as a blank by using the MISSING statement or the MISSING= system option. For more information about the MISSING statement or the MISSING= option, see *SAS Language Reference: Dictionary*. △

When the Pointer Moves Past the End of a Line In a DATA step that uses ODS, the number of columns in the buffer and in the data component are determined in one of three ways:

- By default, the number of variables in the program data vector determines the number of ODS columns.
- You can override the default by defining ODS columns with the COLUMNS= suboption in the FILE PRINT ODS statement.
- If you associate a template with the data component, then the specifications in the template take precedence and might change the number of columns that actually appear in the output object.

When using pointer controls and the @ or @@, you might inadvertently position the pointer beyond the last ODS column. You can control how SAS handles this situation with options in the FILE PRINT ODS statement. For more information see the discussion of overflow control on page 68.

See Also

Statement:

"FILE Statement for ODS" on page 68

Chapter 3, "Output Delivery System and the DATA Step," on page 39

"Examples" on page 41

ODS _ALL_ CLOSE Statement

Closes all open ODS output destinations.

Valid: anywhere

Category: ODS: Output Control

Syntax

ODS _ALL_ CLOSE;

Details

The ODS _ALL_ CLOSE statement closes all open ODS output destinations.

Note: Be sure to open one or more ODS destinations before you execute your next program so that you can view or print your output within the same SAS session. △

ODS CHTML Statement

Opens, manages, or closes the CHTML destination, which produces a compact, minimal HTML that does not use style information.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS CHTML<(<ID=>*identifier*)> <*action*>;

ODS CHTML <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS CHTML statement without an action or options, then it opens the CHTML destination and creates CHTML output.

Actions

The following table lists the actions available for the ODS CHTML statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.1 ODS CHTML Action Summary Table

Task	Action
Close the CHTML destination and the file that is associated with it	CLOSE
Exclude output objects from the CHTML destination	EXCLUDE
Select output objects for the CHTML destination	SELECT
Write to the SAS log the current selection or exclusion list for the CHTML destination	SHOW

Options

The following table lists the options that are available for the ODS CHTML statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.2 ODS CHTML Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the CHTML destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the CHTML destination and specify the file that contains a table of contents for the output	CONTENTS=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify a cascading style sheet to apply to your output	CSSSTYLE=

Task	Option
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the CHTML files that the destination writes to	METATEXT=
Create a new body file at the specified starting point.opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specifies tagset-specific suboptions and a named value	OPTIONS
Specifies that the output from the destination be added to an ODS package	PACKAGE
Open the CHTML destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the CHTML destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=

Task	Option
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS CHTML statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

ODS CSVALL Statement

Opens, manages, or closes the CSVALL destination, which produces HTML output containing columns of data values that are separated by commas, and produces tabular output with titles, notes, and bylines.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS CSVALL < (<ID=>*identifier*)> <*action*>;

ODS CSVALL <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS CSVALL statement without an action or options, then it opens the CSVALL destination and creates CSVALL output.

Actions

The following table lists the actions available for the ODS CSVALL statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.3 ODS CSVALL Action Summary Table

Task	Action
Close the CSVALL destination and the file that is associated with it	CLOSE
Exclude output objects from the CSVALL destination	EXCLUDE

Task	Action
Select output objects for the CSVALL destination	SELECT
Write to the SAS log the current selection or exclusion list for the CSVALL destination	SHOW

Options

The following table lists the options that are available for the ODS CSVALL statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.4 ODS CSVALL Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the CSVALL destination and specify the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the CSVALL destination and specify the file that contains a table of contents for the output	CONTENTS=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE

Task	Option
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the CSVALL destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the CSVALL destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS CSVALL statement is part of the ODS markup family of statements. ODS statements in the markup family open the markup destination and produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

ODS DECIMAL_ALIGN Statement

Controls the justification of numeric columns when no justification is specified.

Valid: anywhere

Category: ODS: SAS Formatted

See: "Values in Table Columns and How They Are Justified" on page 754

Interaction: The ODS DECIMAL_ALIGN statement only effects the RTF destination and the printer family of destinations.

Default: ODS NO_DECIMAL_ALIGN

Syntax

ODS DECIMAL_ALIGN | NO_DECIMAL_ALIGN;

ODS DECIMAL_ALIGN
aligns values by the decimal point in numeric columns when no justification is specified.

Alias: ODS DECIMAL_ALIGN=YES

ODS NO_DECIMAL_ALIGN
right justifies numeric columns when no justification is specified.

Alias: ODS DECIMAL_ALIGN=NO

Details

The ODS DECIMAL_ALIGN statement has no effect on any column that is assigned a justification from a procedure or column definition.

ODS DOCBOOK Statement

Opens, manages, or closes the DOCBOOK destination, which produces XML output that conforms to the DocBook DTD by OASIS.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS DOCBOOK < (<ID=>*identifier*)> <*action*>;

ODS DOCBOOK <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS DOCBOOK statement without an action or options, then it opens the DOCBOOK destination and creates XML output.

Actions

The following table lists the actions available for the ODS DOCBOOK statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.5 ODS DOCBOOK Action Summary Table

Task	Action
Close the DOCBOOK destination and the file that is associated with it	CLOSE
Exclude output objects from the DOCBOOK destination	EXCLUDE
Select output objects for the DOCBOOK destination	SELECT
Write to the SAS log the current selection or exclusion list for the DOCBOOK destination	SHOW

Options

The following table lists the options that are available for the ODS DOCBOOK statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.6 ODS DOCBOOK Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the XML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the XML output	CHARSET=
Open the DOCBOOK destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the DOCBOOK destination and specify the file that contains a table of contents for the output	CONTENTS=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=

Task	Option
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify XML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify XML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the DOCBOOK files that the destination writes to	METATEXT=
Create a new body file at the specified starting point.opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the DOCBOOK destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the DOCBOOK destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=

Task	Option
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS DOCBOOK statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. SAS supplies many markup languages for you to use ranging from DOCBOOK to TROFF. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

ODS DOCUMENT Statement

Opens, manages, or closes the DOCUMENT destination, which produces a hierarchy of output objects that enables you to produce multiple ODS output formats without rerunning a PROC or DATA step.

Valid: anywhere

Category: ODS: Output Control

Syntax

ODS DOCUMENT *action*;

ODS DOCUMENT

<**NAME=**<*libref.*>*member-name* <(*access–option*)>>

<**DIR**=(<**PATH**=*path*<(*access-option*)> <**LABEL**=“*label*”>>)>

<**CATALOG**=*permanent-catalog* | _NULL_>;

Actions

An *action* is any one of the following:

CLOSE
closes the destination and any files that are associated with it.

Tip: When an ODS destination is closed, ODS does not send output to that destination. Closing an unneeded destination frees some system resources.

EXCLUDE *exclusion(s)* | ALL | NONE
excludes one or more output objects from the DOCUMENT destination.

Default: NONE

Restriction: The DOCUMENT destination must be open for this action to take effect.

Main discussion: "ODS EXCLUDE Statement" on page 110

SELECT *selection(s)* | ALL | NONE
selects one or more output objects for the DOCUMENT destination.

Default: ALL

Restriction: The DOCUMENT destination must be open for this action to take effect.

Main discussion: "ODS SELECT Statement" on page 264

SHOW
writes the current selection or exclusion list for the destination to the SAS log.

Restriction: The destination must be open for this action to take effect.

Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.

See also: "ODS SHOW Statement" on page 277

Options

CATALOG=*permanent-catalog* | _NULL_

CAUTION:
If you do not specify a value (other than _NULL_) for this option, then you can replay temporary GRSEGs only during the session in which they are created, not in subsequent sessions. △

permanent-catalog
copies any temporary GRSEG to the specified permanent catalog and keeps a reference to the permanent GRSEG in the document. This value persists until the ODS DOCUMENT statement is closed, or until you delete it by specifying CATALOG=_NULL_.

The *permanent catalog* has the following form:

<*libref.*><*member-name*>;

NULL
deletes the catalog name that was previously specified for the CATALOG= option. Thereafter, temporary GRSEGs are not copied into the permanent catalog, and thus are unavailable in subsequent sessions.

Alias: CAT=

Default: By default, no value is assigned to CATALOG=, which means that temporary GRSEGs are not copied to a permanent catalog.

DIR=

(<PATH=*path* <(*access-option*)>> <LABEL='label'>);
specifies the directory path and/or label for ODS output.

LABEL=*label*
assigns a label to a path.

Requirement: The label that you assign must be enclosed in quotation marks.

Interaction: If LABEL= is used with the PATH= option, then the label applies to the path. If LABEL= is used without the PATH= option, then the label applies to the entire document.

PATH=

path <(*access-option*)>
is specified as a sequence of entries that are delimited by backslashes.

path
can have the form:

path<*#sequence–number*>

where

path
is the name of the path.

#sequence–number
is a number which, when combined with a pathname, uniquely identifies the entry in the directory that contains it.

Default: The default path is "\" (root).

Tip: You can specify a directory that contains entries that do not exist in the document.

access-option
specifies the access mode for the ODS document.

WRITE
opens a document and provides write access as well as read access.

Caution: If the ODS document already exists, then it will be overwritten.

Interaction: If a label has been specified with the LABEL= option, then it will override any existing label assigned to the document.

Tip: If the ODS document does not exist, then it will be created.

UPDATE
opens an ODS document and appends new content to the document. UPDATE provides update access as well as read access.

Caution: If the document already exists, then its contents will not be changed.

Interaction: If a label has been specified with the LABEL= option, then it will be assigned to the document.

Tip: If the ODS document does not exist, then the document will be created.

Default: UPDATE

Note: Procedure output or data queries will be added at the end of the directory. △

NAME=

<*libref.*>*member-name*<(access-option)>

libref
specifies the SAS library where the document is stored.

Default: If no library name is specified, the WORK library is used.

member-name
specifies the document name.

Default: If no NAME= is specified, the specified options apply to the currently open document.

Default: If you do not specify an *access-option* with NAME=, then your directories will open in UPDATE mode.

access-option
specifies the access mode for the ODS document.

WRITE
opens a document and provides write access as well as read access.

Caution: If the ODS document already exists, then it will be overwritten.

Interaction: If a label has been specified with the LABEL= option, then it will override any existing label assigned to the document.

Tip: If the ODS document does not exist, then it will be created.

UPDATE
opens an ODS document and appends new content to the document. UPDATE provides update access as well as read access.

Caution: If the document already exists, then its contents will not be changed.

Interaction: If a label has been specified with the LABEL= option, then it will be assigned to the document.

Tip: If the ODS document does not exist, then the document will be created.

Default: UPDATE

Interaction: If you use the NAME= option in an ODS DOCUMENT statement without closing any instances of the DOCUMENT destination that are already open, the option will force ODS to close the destination and all files associated with it, and to open a new instance of the destination.

ODS ESCAPECHAR Statement

Defines a representative character to be used in output strings.

Valid: anywhere

Category: ODS: Output Control

Restriction: Affects all open destinations except for the LISTING destination.

Restriction: SAS/GRAPH does not support inline formatting.

Syntax

ODS ESCAPECHAR= '*escape-character*';

Required Arguments

escape-character
specifies the special character that identifies the inline formatting symbol. The *escape-character* should be one of the following rarely used characters: `@`, `^`, or `\`.

Note: For RTF output, the `~`, `*`, or `#` can also be used. The `\` is a special RTF character. Therefore, it is recommended that you use an escape character other than `\` for RTF output. △

Note: There is no default value for the escape character, but you can use the special escape sequence **(*ESC*)** in the same way as an escape character. △

With the ODS ESCAPECHAR statement, you can define an escape character for use with the inline formatting functions. These functions provide the ability to enhance and interpret text strings that are used by statements and variables. You can use these functions to modify text strings within table cells and title and footnotes.

Refer to "Using the ODS ESCAPECHAR Functions" on page 101 for a complete list of the inline formatting functions and a detailed description.

Details

Basic Inline Formatting

Inline formatting functions enable you to change styles in title and footnotes, text strings, and table cells. You can insert subscript or superscript into SAS output, underline text, justify text, insert page X of Y numbers into output, insert line feeds into long text strings, and insert destination-specific, raw text into HTML or RTF output.

Existing style elements and style attributes can be used with the STYLE functions. See the explanation of using styles in "Inline Style Attributes and Nesting" on page 98.

Inline Style Attributes and Nesting

You can use the ODS ESCAPECHAR statement and inline formatting syntax to justify text or change the color of titles, footnotes, and text. Other style attributes that are useful and that can add emphasis are font size, underlining, overlining, and strikethrough.

You can change the styles, too. For example, ODS PRINTER now supports two style settings for underlining. ODS PRINTER recognizes the SAS/GRAPH syntax UNDERLIN=1,2,3 for underlining text. However, this option does not change the thickness of the line. The new style element in SAS 9.2 is TextDecoration, which allows you to set the underline, overline, or strike-through styles on titles, footnotes, and text strings. Refer to Appendix 4, "ODS Style Elements," on page 905 and "Style Attributes and Their Values" on page 498 for more details on these functions. See the global title and footnote style options in the TITLES and FOOTNOTE statements in *SAS Language Reference: Dictionary*.

Nested inline formatting is also supported. This feature enables you to set several style attributes for a string without resetting the previously used style. You can start a string with one set of style attributes and add to them later in the string. The first thing to notice is the new syntax:

^{style <style-element-name><[style-attribute-specification(s)]> formatted text}

The syntax begins with the function name STYLE. Next you can add a style element like Headerfixed, SystemTitle, and so on, as needed. Then you can add new attributes such as fontstyle, color, and so on, within brackets. The syntax ends with the text you want to format. The following code is a good example of nested formatting for RTF output:

```
title "test of ^{super ^{style [color=red] red ^{style [color=green] green} and
 ^{style [color=blue] blue }formatting }} and such" ;
```

In the example code, the *^{super<text>* is invoked to start using the superscript function. Then the style function is used to add another style attribute, *^{style [color=red]<text>* for your text.

To understand this nesting, think of a stack in which the first item that is placed on the stack is the last item to come off of the stack (FILO). The superscript function is pushed onto the stack first. Then the style function is used to push a red color onto the stack, resulting in a text string that is superscripted and red in color. Next, the green color is pushed onto the stack. Because the new style attribute is a color, the text changes to the new color. Continuing the string processing, this style attribute is then closed with a close bracket. After the green color is closed or popped off the stack, the red color becomes the active style attribute for the text. Next the blue color is pushed onto the stack, and the text string uses that color. After the blue color is closed, the red and superscript are closed. Now the stack is empty, so ODS uses the default style attributes to finish processing the text string.

Note: Each output destination has limitations. When you use the PRINTER destination, you can only nest styles. The SUB and SUPER functions cannot be nested with the STYLE function. However, the HTML and RTF destinations can nest the STYLE function with the SUB and SUPER functions. △

Using Unicode Symbols

ODS now supports the ability to incorporate Unicode symbols such as Greek symbols into your output. The new inline formatting function is UNICODE, and the syntax is ^*{unicode < value>)*. The syntax is similar to other inline formatting functions in that you can use a four-digit Unicode value that is predefined in a list. Another way to use the UNICODE function is to predefine a list that is stored as a tagset. See "Concepts: Markup Languages and the TEMPLATE Procedure" on page 838 for information on tagsets.

There is a new tagset that contains a predefined list of common Greek symbols and their Unicode values. You can update this template as needed. The template increases the flexibility of the new inline style function. You can still use existing inline style functions, ^{dagger} and ^{sigma}.

To find out what symbols are available to you for Windows XP, you can select **Start ► Programs ► Accessories ► System Tools ► Character Map**. The window that appears shows you the font and all of the symbols available for a given font. For the font displayed, you can highlight a symbol and see the Unicode value for that symbol. The Unicode value is displayed at the bottom of the Character Map window. You can use that Unicode value in the argument to the UNICODE function. For example, Unicode value 216b displays a Roman Numeral 12. The following code illustrates the use of this value:

```
title 'Roman Numeral twelve is ^{unicode 216b}';
```

The Base.Template.Tagsets tagset contains a table of Unicode values and their mnemonics. To add or change a mnemonic, you must SOURCE the tagset, make the desired changes, and then run the modified tagset. The following code creates a file called core.tpl in the current directory that contains the Base.Template.Tagsets tagset also:

```
proc template;
 source base.template.tagset. / file="core.tpl";
run;
```

If you open the core.tpl file, you notice some text that appears similar to the following text:

```
set $unicodeMap["ALPHA" ] "03B1";
set $unicodeMap["BETA" ] "03B2";
set $unicodeMap["DAGGER" ] "2020";
```

To update the file with a new mnemonic and corresponding Unicode value, use the following syntax and add it to the file:

set $unicodeMap["<new function name>"]"<unicode value";

Save the file and compile the modified tagset using PROC TEMPLATE as follows:

```
proc template;
     %inc "core.tpl";
run;
```

The modified tagset is stored in the first writable template store in your ODS path. See "Concepts: Markup Languages and the TEMPLATE Procedure" on page 838 for more information on PROC TEMPLATE and using tagsets.

A new registry setting holds the Unicode font value. You can change this Unicode font value to any valid font that is installed on your computer and recognized by SAS. Refer to "Changing SAS Registry Settings for ODS" on page 32 for specific details on how to change your font. Refer to "Using TrueType Fonts with Universal Printers and SAS/GRAPH Devices in SAS 9.2" in *SAS Language Reference: Concepts* for information on all the new True Type fonts available in SAS 9.2. This chapter contains information on how to install the TrueType fonts on your computer, too.

Inline Formatting With the PUT Statement

Inline formatting information that is used in the PUT statement is counted as printed space that is needed for the ODS Listing output destination. Therefore, the LINESIZE= system option might need to be set to prevent wrapping of the output. For example, the inline formatting information shown in the following code, defines the font size, font face, and font weight for the 'AAA' and 'BBB' values.

```
ods escapechar="^";
ods html file='file.html';
ods pdf file='file.pdf';
ods rtf file='file.rtf';
data _null_;
 file print;
put @1 '^{style [fontsize=8pt] ^{style [fontface=courier]
    ^{style [fontweight=bold]}}}'  'AAA';
put +5 '^{style [fontsize=8pt]  ^{stylefontface=courier]
    ^{style [fontweight=bold]}}}' 'BBB';
run;
ods _all_ close;
```

The line size needed for printed output is three characters. However the inline formatting information is also counted as part of the line size, even though that count only affects the appearance of the output. This increased line size can cause the text to wrap if it exceeds the current value of the LINESIZE= system option.

To prevent wrapping in the ODS Listing output, increase the value of the LINESIZE= system option or decrease the font size.

Inline Formatting With ODS Statistical Graphics

ODS Statistical Graphics include template-based procedures (SGPLOT, SGPANEL, and SGSCATTER) and some statements that support ODS ESCAPECHAR in conjunction with the UNICODE, SUB, and SUP inline formatting functions. Refer to *SAS/GRAPH: Statistical Graphics Procedures Guide* and *SAS/GRAPH: Graph Template Language User's Guide* for information on how to use these functions with ODS Statistical Graphics.

Interpreting Inline Formatting Output Strings

ODS ESCAPECHAR controls the interpretation of output strings by ODS, except for the LISTING, the OUTPUT, and the DOCUMENT destinations. Whenever ODS destinations encounter the specified character in their output, regardless of the source of the output, ODS interprets the character as a special "escape" character that enables special formatting options. For example, the following program produces output in *italics*.

```
data italic;
   x='This font is ^{style[fontstyle=italic]italic}.';
   output;
   run;
ods listing close;
ods pdf file="italicFont.pdf";
ods escapechar='^';
proc print data=italic;
run;
ods _all_ close;
```

Using the ODS ESCAPECHAR Functions

This section describes the use of the defined ODS ESCAPECHAR character value to perform inline formatting. After you define the ODS ESCAPECHAR character value, you can use the available inline formatting functions to change the style within cells, text, titles, and footnotes. You can use these functions to insert page numbers, line feeds, destination-specific raw data, additional spaces and formatting into your output.

Note: For traditional RTF output, you must use **`Print Preview`** to view your output. Some of the formatting will not show up in the SAS results viewer window. △

The inline functions available are shown in Table 5.7 on page 102. Here is the syntax for the ODS ESCAPECHAR functions:

escape-character{*function-name*, <<*arg-1* <*arg-2*…<*arg-n*>>>>}

escape-character
is the character defined using the ODS ESCAPECHAR statement.

{ }
enclose the inline formatting grouping characters.

function-name
is the name of the inline formatting function.

Table 5.7 Valid Functions That Can Be Used with ODS ESCAPECHAR

Function Name	Argument
DAGGER	None
DATE	None
DEST	OUTPUT destination
LASTPAGE	None
LEADERS	String
NBSPACE	Optional number
NEWLINE	Optional number
PAGEOF	None
RAW	String
SIGMA	None
STYLE	Style elements, style attributes, and style= option formats
SUB	Arguments to subscript
SUPER	Arguments to superscript
THISPAGE	None
TOCENTURYINDENT	Length
TOCENTURYPAGE	None
UNICODE	Unicode value

arg-1, arg-n
arguments that are given to the function. The number of arguments depends on the function. Some functions have no arguments.

CAUTION:
A space between the escape character and left bracket of the inline formatting style function will produce undesired results. Here is an example of how the code should look: "^{style [color=green] title green}"; △

DAGGER Function

^{**DAGGER**}
produces the Greek dagger sign.
Tip: It is preferable to use the UNICODE function to generate a dagger sign.
Featured in: Example 1 on page 107

DATE Function

^{**DATE**}
inserts the RTF to express the date.
Tip: You can use this function only with the TAGSETS. RTF destination.

DEST Function

^{**DEST** <[*output-destination*] > text}

output-destination
is one of the ODS output destinations, RTF, printer family, or HTML. This is the destination that will be used by the **`inline-formatting-function`**.

Tip: You can specify more than one *output-destination*.

text
is text that you want to output. An example is:

```
^{dest [rtf html] ^{raw rawtext string} };
```

LASTPAGE Function

^{**LASTPAGE**}

inserts the total number of pages.

Tip: This function works with the PRINTER, RTF, and TAGSETS. RTF destinations.

Tip: You must use Print Preview to view the resolved LASTPAGE function output that is generated by the TAGSETS.RTF destination.

LEADERS Function

^{**LEADERS** <*string*>}

string
is the string that is repeated to fill the space between the leading text and the following text. This function is often used when generating Table of Contents. Example code is:

```
PostText = " ^{leaders .  }^{tocentrypage}
```

Tip: You can use this function only with the PRINTER destination.

NBSPACE Function

^{**NBSPACE** (<*number*>)}

number
is the number of spaces that you want to insert. A single space is inserted if you do not specify a number argument.

Default: The NBSPACE value defaults to 1. A single space is inserted if a number is not specified.

Featured in: Example 1 on page 107

NEWLINE Function

^{**NEWLINE** (<*number*>)}

number
is the number of lines that you want to insert.

Default: The newline value defaults to 1. A single line is inserted if you do not specify a number.

Featured in: Example 1 on page 107

PAGEOF Function

^{**PAGEOF**}
inserts RTF syntax to express all the controls for Page X of Y.

Tip: You can use the PAGEOF function in the TITLE and FOOTNOTE statements. However, if the BODYTITLE option is also specified with the ODS RTF statement, the "page of" information is not written as expected because the BODYTITLE option removes the titles and footnotes from the header and footer sections of the RTF file. If the desired location of the page numbering is in the title or the footnote, remove the BODYTITLE option.

Tip: When the \ character is specified as the ODS ESCAPECHAR character, the PAGEOF function will not be interpreted properly for the TAGSETS.RTF destination. Instead, specify a different escape character.

Tip: You can use the PAGEOF function only with the RTF and TAGSETS. RTF destinations. You must use Print Preview to view the resolved PAGEOF function output that is generated by the TAGSETS.RTF destination.

RAW Function

^{**RAW** <*string*>}

string
is inserted directly without translation. This function allows you to insert control characters. This function works for markup destinations like HTML and RTF.

Restriction: The RAW function does not work with PDF or Window's drivers.

Tip: After this function has been turned on in a session, you cannot turn it off for that session.

Tip: The **** is a special RTF character. The **{** and **}** are special function characters. When you use these special characters with the RAW function, ODS might generate unexpected output.

Featured in: Example 1 on page 107

SIGMA Function

^{**SIGMA**}
generates the Greek SIGMA sign σ.

Tip: The preferable way to produce a SIGMA sign is to use the UNICODE function.

Featured in: Example 1 on page 107

STYLE Function

^{**STYLE** <*style-element-name*> <[*style— attribute-specification*] > formatted text}

style-element-name
specifies the style element. For the STYLE function, you can use the same format that is available for the STYLE= option in all of the templates. For example:

```
^{style rowheader [color=red] my text};
```

or

```
^{style rowheader my text};
```

See: Appendix 4, "ODS Style Elements," on page 905

style-attribute-specification
specifies the style attribute. For the STYLE function, you can use the same format that is available for the STYLE= option in all the templates. For example:

```
^{style [color=red] my text};
```

See: The list of Style Attributes and their values in "Style Attributes and Their Values" on page 498

formatted-text
specifies the text to which to apply the styles.

Note: The style attributes or elements remain in effect until they are overridden by another style. A default style can also reset the style. The following code illustrates that the bolded text style is reset by the sub functions default style.

```
ods pdf text='^{style [fontweight=bold] BOLDED} ^{sub a} NOT bolded}'
```

△

Note: You can nest inline styles. The following example illustrates nesting inline styles.

```
^{style [color=red] red  ^{style [fontsize=18pt] big}}
```

Refer to "Inline Style Attributes and Nesting" on page 98 for an explanation of nesting styles. △

SUB Function

^{**SUB** <*subscript-character*>}

subscript-value
can be a numeric, alphanumeric, or a character value. This value is written below and immediately to one side of another character.

Restriction: Microsoft Word honors only one level of subscript for RTF and TAGSETS RTF.

Restriction: The PRINTER destination does not recognize nesting of the SUB function. The *subscript-value* must immediately follow the SUB function.

Featured in: Example 1 on page 107

SUPER Function

^{**SUPER** <*superscript-value*>}

superscript-value
can be a numeric, alphanumeric, or a character value. This value is written above and immediately to one side of another character.

Restriction: Microsoft Word only honors one level of subscript for RTF and TAGSETS.RTF.

Restriction: Nesting of the SUPER function is not recognized by the PRINTER destination. The *subscript-value* must immediately follow the SUPER function.

Featured in: Example 1 on page 107

THISPAGE Function

^{**THISPAGE**}
inserts the current page number.

Tip: This function can be used only with the PRINTER, RTF, and TAGSETS.RTF destinations.

Tip: You must use Print Preview to view the resolved THISPAGE function output that is generated by the TAGSETS.RTF destination.

TOCENTRYINDENT Function

^{**TOCENTRYINDENT** <*len*>}

len
is the amount to be indented per level. Example code is:

```
PreText = "^{tocentryindent 2em}"
```

Tip: This function can be used only with the PRINTER destination.

TOCENTRYPAGE Function

^{**TOCENTRYPAGE**}

is the page number of the current TOC entry. Example code is:

```
PostText = " ^{leaders .  } ^{tocentrypage}"
```

Tip: This function can be used only with the PRINTER destination.

UNICODE Function

^{**UNICODE** <*unicode-value* | *'unicode-value'X*>}

unicode-value
can be an actual four-place hexadecimal Unicode value or one of the names listed in the Base.Template.Tagsets template. For example, 03B2 is the Unicode value for the Alpha symbol. Refer to "Using Unicode Symbols" on page 99 for details about Unicode values.

Tip: Thorndale Duospace WT J is the default font used in the inline style Unicode function for the PDF destination.

Featured in: Example 1 on page 107

'unicode-value'X
is syntax used with STAT/GRAPH. A hexadecimal value is enclosed in single or double quotes followed by an **x**. The **x** specifies that the value in quotes is a hexadecimal value. This quoted value must be an actual four-place hexadecimal Unicode value or one of the names listed in the Base.Template.Tagsets template. For example, 03B2 is the Unicode value for the Alpha symbol. Refer to "Using Unicode Symbols" on page 99 for details about Unicode values.

Tip: Thorndale Duospace WT J is the default font used in the inline style Unicode function for the PDF destination.

Tip: The *unicode-value* can be enclosed with single or double quotes.

Example

Example 1: Basic Inline Formatting Functions

ODS features:

ODS RTF statement:

Action:

CLOSE

Options:

FILE=

Other SAS features:

OPTIONS statement

PROC PRINT

TITLE statement

Program Description The following example highlights inline formatting functions that are supported for all destinations. It also shows how to nest inline formatting functions. In this example, RTF is the destination used.

Note: To see all of the styles and colors displayed properly, use **`Print Preview`** to view the output. △

Program

Turn off the Date and Page number. The NODATE option turns off the output of the date and time, and the NONUMBER option tells SAS not to print the page number on the first title line of each page of output.

```
options nodate nonumber;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Set the escape character for inline formatting.

```
ods escapechar="^";
```

Create RTF and PDF output. The ODS RTF statement opens the RTF destination and creates RTF output. The ODS PDF statement opens the PDF destination and creates PDF output.

```
ods rtf file="rtfInlinFuncs.rtf";
ods rtf file="pdfInlinFuncs.pdf";
```

Set the TITLE statement. This TITLE statement provides the topic title for the RTF output.

```
title "Examples of Inline Formatting Functions";
```

Show the NBSPACE function. The non-breaking spaces function, NBSPACE, puts the number of spaces that you specified in the output of the title.

```
title2 'Example of ^{nbspace 3} Non-Breaking Spaces Function';
```

Show the NEWLINE function. The NEWLINE function puts the specified number of additional line feeds in the output of the title.

```
title3 'Example of ^{newline 2} Newline Function';
```

Show the RAW function. The RAW function puts the escaped text that you specified into the file exactly as it is shown. Each ODS destination has special instructions that are recognized. In the following code **\cf12** is an instruction that the RTF destination recognizes and can display. The PDF destination does not recognize this instruction.

```
title4 'Example of ^{raw \cf12 RAW} RAW function';
```

Show the UNICODE function. This TITLE statement shows how the UNICODE function works.

```
title5 'Example of ^{unicode 03B1} UNICODE function';
```

Show the STYLE function and the nesting of functions. This TITLE statement shows the STYLE function using style attribute FOREGROUND=. This example also shows the nesting of the STYLE, SUPER, and UNICODE functions.

```
title6 "Example ^{style [foreground=red] of Super, Alpha ^{super ^{unicode ALPHA}
       ^{style [foreground=green] Nested}} Formatting} and Scoping";
```

Show the SUPER function and the nesting of functions. This TITLE statement shows the STYLE function using style attribute FOREGROUND=. This example also shows the nesting of the STYLE, SUB, and SIGMA functions.

```
title7 "Example of SUB, ^{sub
         ^{style [foreground=red] red
         ^{style [foreground=green] green } and
         ^{style [foreground=blue] blue styles }}} and SIGMA Functions";
```

Print the DATA set.

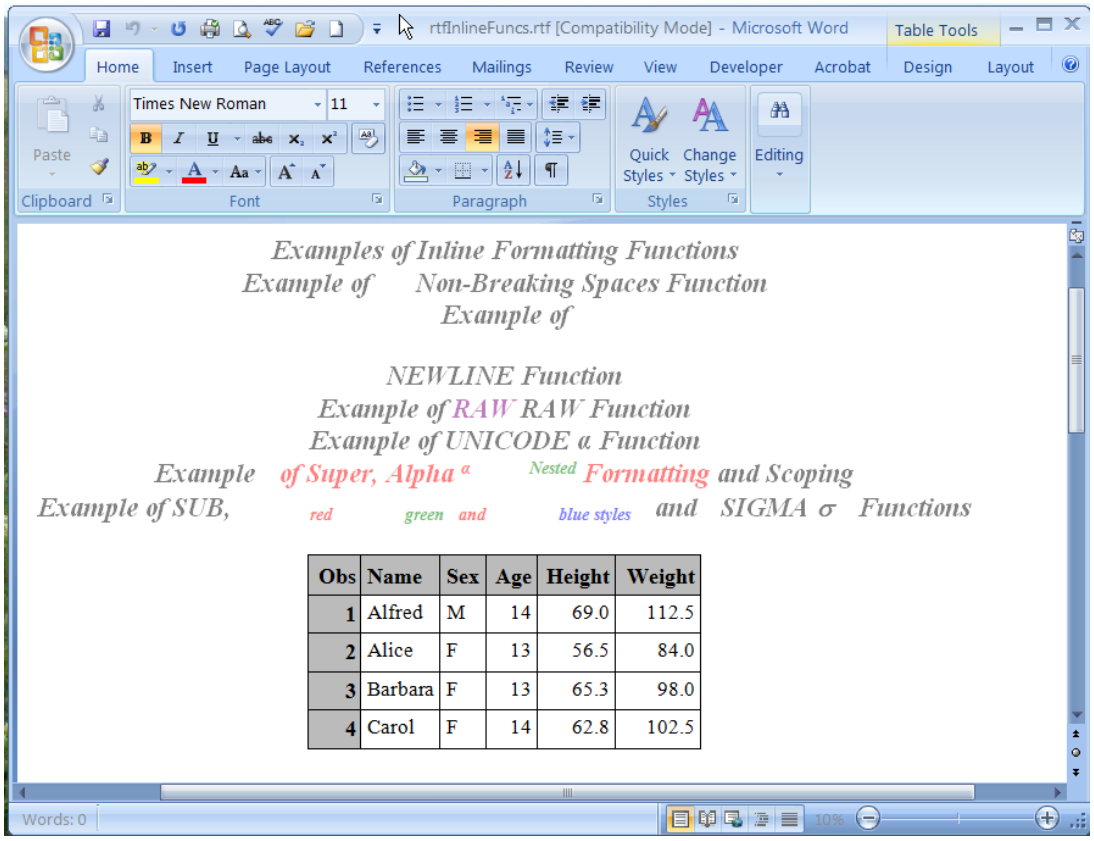
```
proc print data=sashelp.class(obs=4);
run;
```

Close the ODS destinations. The ODS _ALL_ CLOSE statement closes the RTF and PDF destinations and all of the files that are associated with it. If you do not close the destination, you cannot view the files in a browser window.

```
ods _all_close;
```

RTF Output

This output shows the basic inline formatting functions and how you can use them with the TITLE statement, starting with the non-breaking line function (NBREAK). The other functions used in the code and shown in the following output are NEWLINE, RAW, UNICODE, ALPHA, STYLE, SUPER, SUB, and SIGMA. Nesting functions are also demonstrated in the RTF output. Note that only one level of nesting occurs with the SUB and SUPER functions in the RTF output.

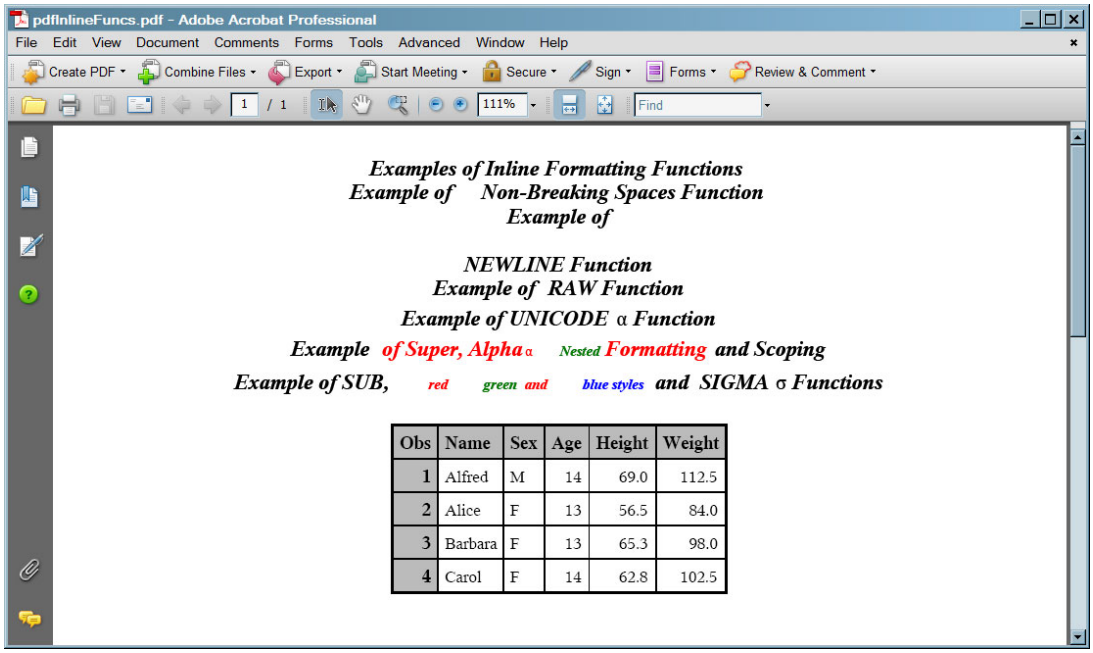


Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5

PDF Output

This output shows the basic inline formatting functions and how you can use them with the TITLE statement, starting with the non-breaking line function(NBREAK). The other functions used in the code and shown in the following output are NEWLINE, RAW, UNICODE, STYLE, SUPER, SUB, and SIGMA. Nesting functions are also demonstrated in this PDF output. Note that the SUB and SUPER functions are not honored when nested in the PDF destination. Notice that the SUPER function is not recognized in **`title6`** because of where it is nested. The PDF destination does not recognize the SUB function properly because the subscript-value does not immediately follow the SUB function.

Also note that in **`title4`**, the PDF destination cannot display the special instruction provided in the RAW function. The **`\cf12`** instruction is an RTF instruction.



Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5

ODS EXCLUDE Statement

Specifies output objects to exclude from ODS destinations.

Valid: anywhere

Category: ODS: Output Control

Syntax

ODS <*ODS-destination*> **EXCLUDE** *exclusion(s)*| ALL | NONE;

Required Arguments

exclusion(s)
specifies one or more output objects to add to an exclusion list.

By default, ODS automatically modifies exclusion lists at the end of a DATA step that uses ODS, or at the end of a procedure step. For information about modifying these lists, see "Selection and Exclusion Lists" on page 34.

Each exclusion has the following form:

output-object <(PERSIST)>

output-object
specifies one or more output objects to exclude. To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. You can specify an output object in any of the following ways:

- a full path. For example,

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  is the full path of the output object.
- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  the partial paths are:

  ```
  City_Pop_90.TestsForLocation
  TestsForLocation
  ```

- a label that is enclosed by quotation marks.

  For example,

  ```
  "The UNIVARIATE Procedure"
  ```

- a label path. For example, the label path for the output object is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```

  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement. △
- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```

  the partial label paths are:

  ```
  "CityPop_90"."Tests For Location"
  "Tests For Location"
  ```

- a mixture of labels and paths.
- any of the partial path specifications, followed by a pound sign (#) and a number. For example, **`TestsForLocation#3`** refers to the third output object that is named **`TestsForLocation`**.

See also: "ODS TRACE Statement" on page 317.

(PERSIST)
keeps the *output-object* that precedes the PERSIST option in the exclusion list until you explicitly modify the list with any of the following ODS statements:

- any ODS SELECT statement
- ODS EXCLUDE NONE
- ODS EXCLUDE ALL
- an ODS EXCLUDE statement that applies to the same output object but does not specify PERSIST

This action is true even if the DATA or procedure step ends.

Requirement: You must enclose PERSIST in parentheses.

ALL
specifies that ODS does not send any output objects to the open destination.

Alias: ODS EXCLUDE DEFAULT

Interaction: If you specify ALL without specifying a destination, ODS sets the overall list to EXCLUDE ALL and sets all other lists to their defaults.

Tip: Using ODS EXCLUDE ALL is different from closing a destination. The destination remains open, but no output objects are sent to it.

Tip: To temporarily suspend a destination, use ODS SELECT NONE. Use ODS SELECT ALL when you want to resume sending output to the suspended destination.

NONE
specifies that ODS send all of the output objects to the open destination.

Interaction: If you specify the NONE argument without specifying a destination, ODS sets the overall list to EXCLUDE NONE and sets all other lists to their defaults.

Tip: ODS EXCLUDE NONE has the same effect as ODS SELECT ALL.

Tip: To temporarily suspend a destination, use ODS SELECT NONE. Use ODS SELECT ALL when you want to resume sending output to the suspended destination.

Options

NOWARN
suppresses the warning that an output object was requested but not created.

ODS-destination
specifies to which ODS destination's exclusion list to write, where *ODS-destination* can be any valid ODS destination. For a discussion of ODS destinations, see "Understanding ODS Destinations" on page 24.

Default: If you omit *ODS-destination*, ODS writes to the overall exclusion list.

Tip: To set the exclusion list for the output destination to something other than the default, use the "ODS OUTPUT Statement" on page 184.

WHERE=*where-expression*
excludes output objects that meet a particular condition. For example, the following statement excludes only output objects with the word "Histogram" in their name:

```
ods exclude where=(_name_ ?  'Histogram');
```

where-expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. *where-expression* has this form:

(*subsetting-variable* <*comparison-operator where-expression-n*>)

subsetting-variable
is a special kind of WHERE expression operand used by SAS to help you find common values in items. For example, this EXCLUDE statement excludes only output objects with the path **City_Pop_90.TestsForLocation** :

```
ods exclude / where=(_path_ = 'City_Pop_90.TestsForLocation' );
```

subsetting-variable is one of the following:

LABEL
is the label of the output object.

LABELPATH
is the label path of the output object.

NAME
is the name of the output object.

PATH
is the full or partial path of the output object.

operator
compares a variable with a value or with another variable. *operator* can be AND, OR NOT, OR, AND NOT, or a comparison operator.
The following table lists some comparison operators:

Table 5.8 Examples of Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Details

You can maintain a selection list for one destination and an exclusion list for another. However, the results are less complicated if you maintain the same types of lists for all the destinations to which you route output.

Example

Example 1: Conditionally Excluding Output Objects and Sending Them to Different Output Destinations

ODS features:

- ODS EXCLUDE statement:
  - *ODS-destination* option
  - WHERE= option
- ODS HTML statement:
  - CONTENTS=
  - FRAME=
  - PAGE=
  - TEXT=
- ODS PDF statement:
  - TEXT=
  - STARTPAGE=

Other SAS features:

- PROC UNIVARIATE

Program

Create the BPressure data set.

```
options nodate;
   data BPressure;
      length PatientID $2;
      input PatientID $ Systolic Diastolic @@;
      datalines;
   CK 120 50  SS 96  60 FR 100 70
   CP 120 75  BL 140 90 ES 120 70
   CP 165 110 JI 110 40 MC 119 66
   FC 125 76  RW 133 60 KD 108 54
   DS 110 50  JW 130 80 BH 120 65
   JW 134 80  SB 118 76 NS 122 78
   GS 122 70  AB 122 78 EC 112 62
   HH 122 82
   ;
   run;
```

Create HTML output and add text.

```
   ods html text='Systolic Blood Pressure' file='Systolic-body.html'
            frame='Systolic-frame.htm'
            contents='Systolic-contents.htm'
            page='Systolic-page.htm';
```

Create PDF output and add text.

```
ods pdf file='Diastolic.pdf' text='Diastolic Blood Pressure' startpage=no;
```

Exclude output objects from different output destinations. The first ODS EXCLUDE statement excludes output objects from the HTML destination that have 'Diastolic' in the path name. The second ODS EXCLUDE statement excludes output objects from the PDF destination that have 'Systolic' in the path name.

```
ods html exclude where=(_path_ ? "Diastolic" ) ;
ods pdf exclude where=(_path_ ? "Systolic" ) ;
```

Create the output objects. As PROC UNIVARIATE sends each output object to the Output Delivery System, ODS does not send the output objects from PROC UNIVARIATE that match the items in the exclusion list to the open destinations.

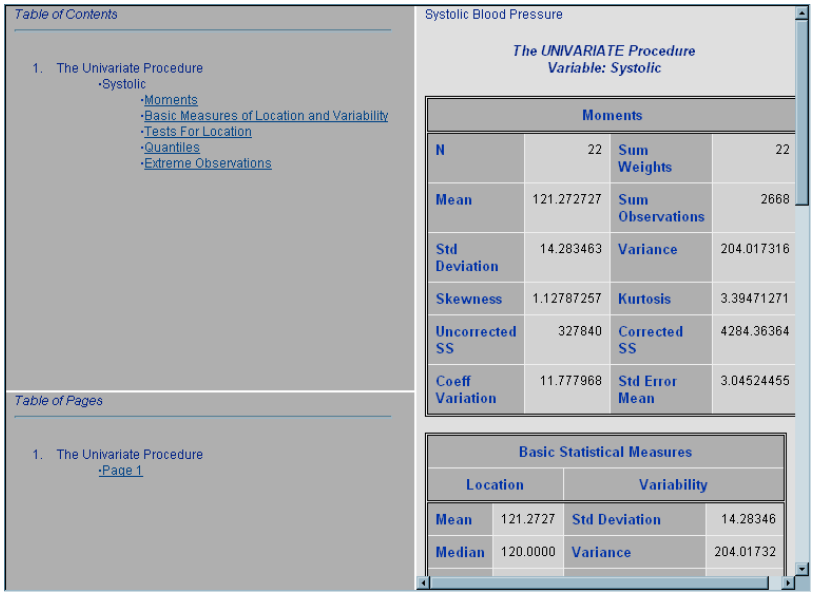
```
proc univariate data=BPressure;
   var Systolic Diastolic;
run;
```

Close the HTML destination. This ODS HTML statement closes the HTML destination and all the files that are associated with it.

```
ods html close;
```

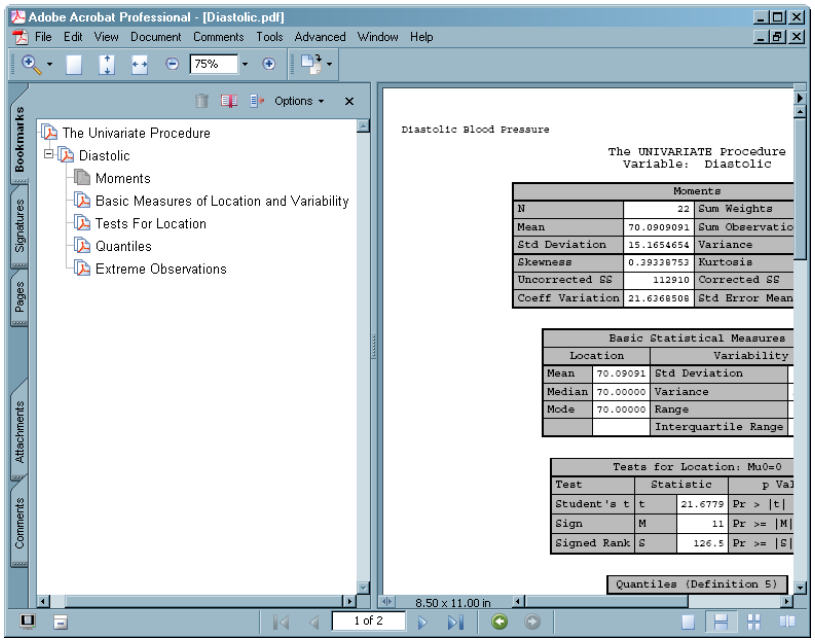
Partial HTML Output

Display 5.3 HTML Output with Systolic Output Objects



Partial PDF Output

Display 5.4 PDF Output with Diastolic Output Objects



See Also

Statements:

"ODS SELECT Statement" on page 264
"ODS SHOW Statement" on page 277
"ODS TRACE Statement" on page 317

ODS GRAPHICS Statement

Enables or disables ODS graphics processing and sets graphics environment options. This statement affects ODS template-based graphics only. The ODS GRAPHICS statement does not affect device-based graphics.

Valid: anywhere

Category: ODS: Output Control

Default: The value of the SAS registry entry "ODS > STATISTICAL GRAPHICS > Default State", which is usually OFF.

Interaction: SAS/GRAPH device-based global statements such as GOPTIONS, SYMBOL, PATTERN, AXIS, and LEGEND do not affect template-based graphics. The ODS GRAPHICS statement does not affect device-based graphics.

Syntax

ODS GRAPHICS <OFF | ON> </ *option(s)*>;

Without Arguments

If the ODS automatic graphic capabilities are currently disabled, then specifying the ODS GRAPHICS statement without options enables them. If the ODS automatic graphic capabilities are currently enabled, then specifying the ODS GRAPHICS statement leaves them enabled.

Required Arguments

ON
enables ODS Graphics processing. This is the default if no argument is used.
Alias: YES

OFF
disables ODS Graphics processing.
Alias: NO

Options

Table 5.9 ODS GRAPHICS Option Summary Table

Task	Option
Specify whether anti-aliasing is applied to the rendering of the line and markers in any graph	ANTIALIAS= | ANTIALIAS | NOANTIALIAS
Specify the maximum number of markers or lines to be anti-aliased before anti-aliasing is disabled	ANTIALIASMAX=
Specify whether to draw a border around each graph	BORDER= | BORDER | NOBORDER
Specify the maximum number of discrete values to be shown in any graph	DISCRETEMAX=
Specify the maximum number of group values to be shown in any graph	GROUPMAX=
Specify the height of any graph	HEIGHT=
Specify the image format used to generate image files	IMAGEFMT=
Specify whether data tips are generated	IMAGEMAP=| IMAGEMAP | NOIMAGEMAP
Specify the base image filename	IMAGENAME=
Specify the maximum number of labeled areas before labeling is disabled	LABELMAX=
Specify an integer that is interpreted as the maximum percentage of the overall graphics area that a legend can occupy	MAXLEGENDAREA=
Specify the maximum number of cells in a graph panel where the number of cells is determined dynamically by classification variables	PANELCELLMAX=

Task	Option
Reset one or more ODS GRAPHICS options to its default	RESET | RESET=
Specify whether the content of any graph is scaled proportionally	NOSCALE | SCALE | SCALE=
Specify the maximum number of distinct mouse-over areas allowed before data tips are disabled	TIPMAX=
Specify the width of any graph	WIDTH=

ANTIALIAS= | ANTIALIAS | NOANTIALIAS
specifies whether anti-aliasing is applied to the rendering of the line and markers in any graph. Anti-aliasing smooths the appearance of diagonal lines and some markers. Text displayed in the graph is always anti-aliased. For graphical displays that plot large numbers of points it is recommended that ANTIALIAS=OFF be specified for performance considerations.

ANTIALIAS= OFF | ON
specifies whether anti-aliasing is applied to the rendering of the line and markers in the graph.

OFF — does not smooth jagged edges of components other than text in the graph.
Alias: NO

ON — smooths jagged edges of all components in the graph.
Alias: YES

ANTIALIAS
smooths jagged edges of all components in the graph.

NOANTIALIAS
does not smooth jagged edges of components other than text in the graph.

Default: ON

Restriction: If the number of markers or curve points in the plot exceeds the number specified by the ANTIALIASMAX= option, then the ANTIALIAS option is turned off, even if you specify the option ANTIALIAS=ON or ANTIALIAS.

ANTIALIASMAX= *n*
specifies the maximum number of markers or lines to be anti-aliased before anti-aliasing is disabled. For example, if there are more than 400 scatterpoint markers to be anti-aliased and ANTIALIASMAX=400, then no markers will be anti-aliased.

n
specifies a positive integer.

Default: 600

BORDER= | BORDER | NOBORDER
specifies whether to draw a border around any graph.

BORDER= OFF | ON
specifies whether to draw the graph with a border on the outermost layout.

ON
specifies to draw a border around the graph.

Alias: YES

OFF
specifies not to draw a border around the graph.

Alias: NO

BORDER
specifies whether to draw a border around the graph.

NOBORDER
specifies not to draw a border around any graph.

Default: BORDER or BORDER=ON

DISCRETEMAX=*n*
specifies the maximum number of discrete values to be shown in any graph. Barcharts and box plots are examples of affected plot types. Scatter plots and other plot types can be affected if the data to be plotted is discrete or the axis is discrete.

n
specifies a positive integer.

Default: 1000

Tip: Some plot layers might be unaffected by the DISCRETEMAX= option, and those layers will still be rendered. If all layers are affected, a blank graph will be rendered.

Tip: If the value specified by the DISCRETEMAX= option is exceeded by any plot layer in the graph, that layer will not be drawn and a warning message is issued.

GROUPMAX=*n*
specifies the maximum number of group values to be shown in any graph. Any graph that supports the GROUP= option is affected.

n
specifies a positive integer.

Default: 1000

Tip: If the value specified by the GROUPMAX= option is exceeded by any plot layer in the graph, that layer will be rendered ignoring the GROUP= option and a warning message is issued.

HEIGHT=*dimension*
specifies the height of any graph.

dimension
is a nonnegative number.

See: *dimension on page 535*

Default: The value of the SAS registry entry "ODS > STATISTICAL GRAPHICS > Design Height" or the value of the DesignHeight= option in a STATGRAPH template. Typically, the value is 480px.

IMAGEFMT= *image–file–type* | STATIC
specifies the image format to be used. If the image format is not valid for the active output destination, the format is automatically changed to the default image format for that destination.

image-file-type
is the image format to be generated. See “Supported Image File Types for Output Destinations” on page 123.

STATIC
uses the best quality static image format for the active output destination. This is the default.

Default: STATIC

IMAGEMAP= | IMAGEMAP | NOIMAGEMAP
controls data tips generation. Data tips are pieces of explanatory text that appear when you mouse-over the data portions of a graph contained in an HTML page.

IMAGEMAP= ON | OFF
controls data tips generation.

OFF — specifies not to generate data tips.
Alias: NO

ON — specifies to generate data tips.
Alias: YES

IMAGEMAP
specifies to generate data tips.

NOIMAGEMAP
specifies not to generate data tips.

Default: OFF or NOIMAGEMAP

Restriction: This option applies only when the ODS HTML destination is used.

IMAGENAME="*filename*"
specifies the base image filename.

If more than one image is generated, each is assigned filename as a base name followed by a number in order to create unique names. This numbering can be reset with the RESET=INDEX option. Path information (if needed) can be set with the GPATH= option on the ODS destination statement. The default path is the current output directory. A file extension for filename is automatically generated based on the IMAGEFMT= option.

Requirement: You must enclose *filename* in quotation marks.

Restriction: *filename* must be a single name. It must not include any path specification or image-format name extension.

Default: The name of the output object.

LABELMAX= *n*
specifies the maximum number of labeled areas before labeling is disabled. For example, if there are more than 50 points to be labeled and LABELMAX=50, then no points will be labeled.

n
specifies a positive integer.

Default: 200

MAXLEGENDAREA= *n*
specifies an integer that is interpreted as the maximum percentage of the overall graphics area that a legend can occupy.

n
specifies a positive integer.

Default: 20

Tip: To turn off the legend, specify MAXLEGENDAREA=0. No warning will be issued when the legend is turned off in this way.

PANELCELLMAX=*n*
specifies the maximum number of cells in a graph panel where the number of cells is determined dynamically by classification variables.

n
specifies a positive integer.

Default: 10000

Tip: Graphs with DataPanel or DataLattice layouts are affected. If the value specified by the PANELCELLMAX= option is exceeded by either of these layouts, an empty graph will be rendered and a warning message is issued.

RESET | RESET= ***option***
resets one or more ODS GRAPHICS options to its default.

RESET
resets all of the *options* to their defaults.

RESET=
resets one of the following to its default:

ALL
resets all of the *reset-options* to their defaults.

ANTIALIAS
resets the ANTIALIAS option to its default.

See also: ANTIALIAS=

ANTIALIASMAX
resets the ANTIALIASMAX option to its default.

See also: ANTIALIASMAX

BORDER
resets the BORDER= option to its default.

See also: BORDER=

IMAGEMAP
resets the IMAGEMAP= option to its default.

INDEX
resets the index counter that is appended to static image files.

HEIGHT
resets the HEIGHT= option to its default.

See also: HEIGHT=

IMAGEMAP
resets the IMAGEMAP= option to its default.

Note: Not all output destinations support this feature. △

See also: IMAGEMAP=

LABELMAX
resets the LABELMAX= option to its default.

See also: LABELMAX=

MAXLEGENDAREA=
resets the LABELMAX= option to its default.

See also: MAXLEGENDAREA=

SCALE
resets the SCALE= option to its default.

See also: SCALE=

TIPMAX
resets the TIPMAX= option to its default.

See also: TIPMAX =

WIDTH=
resets the WIDTH= option to its default.

SCALE= | SCALE | NOSCALE
specifies whether the content of any graph is scaled proportionally.

NOSCALE
does not scale the components of graph proportionally.

SCALE
scales the components of graph proportionally.

SCALE=
specifies whether the content of the graph is scaled proportionally.

OFF	does not scale the components of graph proportionally. **Alias:** NOSCALE **Alias:** NO
ON	scales the components of graph proportionally. **Alias:** YES

Default: ON or SCALE

TIPMAX=*n*
specifies the maximum number of distinct mouse-over areas allowed before data tips are disabled. For example, if there are more than 400 points in a scatterplot, and TIPMAX=400, then no data tips will appear.

n
specifies a positive integer.

Default: 500

WIDTH=*dimension*
specifies the width of any graph.

dimension
is a nonnegative number.

Default: The value of the SAS registry entry "ODS > STATISTICAL GRAPHICS > Design Width" or the value of the DesignWidth= option in a STATGRAPH template. Typically, this value is 640px.

See: dimension on page 535

Details

Using the ODS GRAPHICS Statement You can enable ODS graphics by using either of the following equivalent statements:

```
ods graphics on;
ods graphics;
```

When you specify one of these statements before your procedure invocation, Base, SAS/STAT, SAS/ETS, and SAS/QC procedures support ODS graphics, either by default or when you specify procedure options for requesting particular graphs.

To disable ODS graphics, specify the following statement:

```
ods graphics off;
```

Note: For SAS/GRAPH procedures that use ODS graphics (SGPLOT, SGPANEL, SGSCATTER, and SGRENDER), ODS graphics is always ON and cannot be disabled. For other products, the initial state ODS graphics is determined by a SAS Registry setting. △

Using the ODS GRAPHICS Statement for Batch Jobs To generate ODS graphics output in UNIX batch jobs, you must set the DISPLAY system option before creating the output. To set the display, enter the following command:

```
export DISPLAY=<ip_address>:0
```

The *ip_address* is the TCP/IP address, or the name of a UNIX terminal. Usually, the IP address of the UNIX system where SAS is running would be used. If you do not set the DISPLAY variable, then you get an error message in the SAS log.

Supported Image File Types for Output Destinations The following table lists all of the supported image file types for ODS output destinations.

Output Destination	Supported Image File Types
HTML	PNG (default), GIF, JPEG, JPG
LISTING	PNG (default), BMP, DIB, EMF, EPSI, GIF, JFIF, JPEG, JPG, PBM, PDF, PS, SASEMF, STATIC, TIFF, WMF
LATEX	PS(default), EPSI, GIF, PNG, PDF, JPG
PRINTER Family	PNG(default), JPEG, JPG, GIF
RTF	PNG(default), JPEG, JPG, JFIF
Markup Tagsets	All Markup family tagsets have the default *imagefmt* value built in.

Description of Supported Image File Types

Table 5.10 Description of Supported Image File Types

Image File Type	Description
BMP (Microsoft Windows Device Independent Bitmap)	Supports color-mapped and true color images that are stored as uncompressed or run-length encoded data. BMP was developed by Microsoft Corporation.
DIB (Microsoft Windows Device Independent Bitmap)	See the description of BMP. DIB is supported only under the OS/2 operating system.
EMF (Microsoft NT Enhanced Metafile)	Supported only under Windows 95, Windows 98, and Windows NT.
EPSI (Microsoft NT Enhanced Metafile)	An extended version of the standard PostScript (PS) format. Files that use this format can be printed on PostScript printers and can also be imported into other applications. Notice that EPSI files can be read, but PS files cannot be read.

Image File Type	Description
GIF (Graphics Interchange Format)	Supports only color-mapped images. GIF is owned by CompuServe, Inc.
JFIF (JPEG File Interchange Format)	Supports JPEG image compression. JFIF software is developed by the Independent JPEG Group.
JPEG or JPG (Joint Photographic Experts Group)	A file format that is used for storing noninteractive images.
PBM (Portable Bitmap Utilities)	Supports gray, color, RGB, and bitmap files. The Portable Bitmap Utilities are a set of free utility programs that were developed primarily by Jef Poskanzer.
PDF (Portable Document Format)	A file format for electronic distribution and exchange of documents.
PNG (Portable Network Graphic)	Supports true color, gray-scale, and 8-bit images.
PS (PostScript Image File Format)	The Image classes use only PostScript image operators. A level II PS printer is required for color images. PostScript was developed by Adobe Systems, Inc.
SASEMF (Enhanced Metafile)	EMF image tuned for RTF output.
STATIC	Chooses the best image format for the current ODS destination.
TIFF (Tagged Image File Format)	Internally supports a number of compression types and image types, including bitmapped, color-mapped, gray-scaled, and true color. TIFF was developed by Aldus Corporation and Microsoft Corporation and is used by a wide variety of applications (available if licensed).
WMF (Microsoft Windows Metafile)	Supported only under MicroSoft Windows operating systems.

ODS HTML Statement

Opens, manages, or closes the HTML destination, which produces HTML 4.0 output that contains embedded style sheets.

Valid: anywhere

Category: ODS: Third-Party Formatted

Restriction: When you open the destination, a style sheet is written and linked to the body file. Therefore, you cannot make style sheet changes from within your SAS program. For example, after the destination is open, changing the value of the STYLE= option has no effect. You can make style changes in either of the following ways:

- Close the destination, edit or create a new style sheet, and submit the program again specifying the new or modified style sheet.
- Edit the body file, changing the style sheet url to the desired style sheet.

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS

Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
```

Operating Environment Information: If you use graphics that are created with either the ACTXIMG or JAVAIMG device drivers in the z/OS operating environment, then specify either the GPATH= or the PATH= option in the ODS HTML statement.

Syntax

ODS HTML <(<ID=>*identifier*)> <*action*>;

ODS HTML <(<ID=>*identifier*)><*option(s)*>;

Without an Action or Options

If you use the ODS HTML statement without an action or options, then it opens the HTML destination and creates HTML output.

Actions

The following table lists the actions available for the ODS HTML statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.11 ODS HTML Action Summary Table

Task	Action
Close the HTML destination and the file that is associated with it	CLOSE
Exclude output objects from the HTML destination	EXCLUDE
Select output objects for the HTML destination	SELECT
Write to the SAS log the current selection or exclusion list for the HTML destination	SHOW

Options

The following table lists the options that are available for the ODS HTML statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.12 ODS HTML Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view ODS HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the HTML destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the HTML destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify a device for the HTML output destination	DEVICE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the HTML files that the destination writes to	METATEXT=

Task	Option
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the HTML destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the HTML destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS HTML statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

Examples

Example 1: Creating a Separate Body File for Each Page of Output

ODS features:
    ODS HTML statement:
        Action:
            CLOSE
        Arguments:
            CONTENTS=

BODY=

FRAME=

PAGE=

Options:

BASE=

NEWFILE=

Other SAS features:

#BYVAL parameter in titles

NOBYLINE|BYLINE system option

OPTIONS statement

PROC FORMAT

PROC SORT

PROC REPORT

PROC TABULATE

TITLE statement

Data set:
See "Creating the Grain_Production Data Set" on page 878.

Format:
See "Creating the $CNTRY Format" on page 869.

Program Description The following example creates a separate HTML file for each page of procedure output, as well as a table of contents, a table of pages, and a frame file. The table of contents and table of pages do not appear any different or behave any differently from those that would be created if all the output were in a single file. Because the output is in separate files, you cannot scroll from one page of output to the next. However, you can select individual HTML files to include in a report.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program

Sort the data set Grain_Production. PROC SORT sorts the data, first by values of the variable Year, then by values of the variable Country, and finally by values of the variable Type.

```
proc sort data=grain_production;
   by year country type;
run;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output.

The FRAME=, CONTENTS=, and PAGE= options create a frame that includes a table of contents and a table of pages that link to the contents of the body file. The body file also appears in the frame. BASE= specifies a string to use as the first part of all links and references to the HTML files. Because no URL is specified for individual files, the final part of the link will match the filename.

CAUTION:
The string that the BASE= option specifies must be a valid path to your HTML files. △

```
ods html body='grain-body.htm'
     contents='grain-contents.htm'
        frame='grain-frame.htm'
         page='grain-page.htm'
         base='http://www.yourcompany.com/local-address/'
```

Specify that SAS create a new body file for each page of output. The NEWFILE=PAGE option opens and creates a new body file for each page of output.

```
     newfile=page;
```

Suppress the default BY line and specify a new value into the BY line. The NOBYLINE option suppresses the default BY line variable. The #BYVAL parameter specification inserts the current value of the BY variable Year into the title.

```
options nobyline;
title 'Leading Grain-Producing Countries';
title2 'for #byval(year)';
```

Produce a report. This PROC REPORT step produces a report on grain production. Each BY group produces a page of output, so ODS creates a new body file for each BY group. The NOWINDOWS option specifies that PROC REPORT runs without the REPORT window and sends its output to the open output destination(s).

```
proc report data=grain_production nowindows;
   by year;
   column country type kilotons;
   define country  / group width=14 format=$cntry.;
   define type     / group 'Type of Grain';
   define kilotons / format=comma12.;
   footnote 'Measurements are in metric tons.';
run;
```

Restore the default BY line and clear the second TITLE statement. The BYLINE option restores the default BY line. The TITLE2 statement clears the second TITLE statement.

```
options byline;
title2;
```

Produce a report. The TABLE statement in this PROC TABULATE step has the variable Year has the page dimension. Therefore, PROC TABULATE explicitly produces one page of output for 1995 and one for 1996. ODS starts a new body file for each page.

```
proc tabulate data=grain_production format=comma12.;
   class year country type;
```

```
   var kilotons;
   table year,
         country*type,
         kilotons*sum=' ' / box=_page_ misstext='No data';
   format country $cntry.;
   footnote 'Measurements are in metric tons.';
run;
```

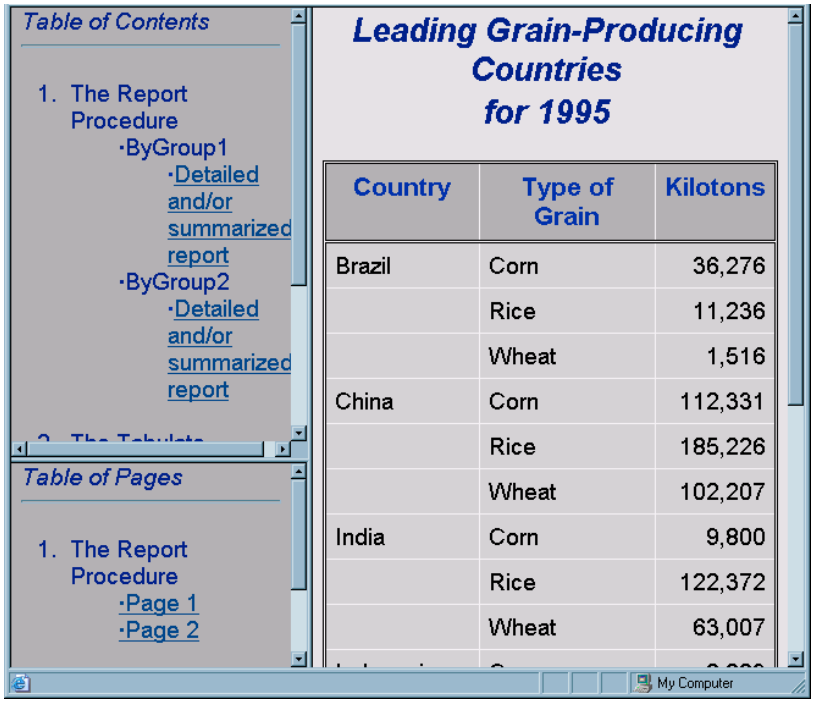
Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files in a browser window.

```
ods html close;
```

HTML Output

Display 5.5 HTML Frame File

This frame file shows the first body file. Links in the table of contents and the table of pages point to the other body files.



Links That Are Created in the HTML Output These HREF= attributes from the links in the contents file point to the HTML tables that ODS creates from the PROC REPORT and PROC TABULATE steps.

```
HREF='http://www.yourcompany.com/local-address/grain-body.htm#IDX'
HREF='http://www.yourcompany.com/local-address/grain-body1.htm#IDX1'
HREF='http://www.yourcompany.com/local-address/grain-body2.htm#IDX2'
HREF='http://www.yourcompany.com/local-address/grain-body3.htm#IDX3'
```

Notice how these HREF attributes are constructed:

- The value of the BASE= option provides the first part of the HREF, which is http://*www.yourcompany.com/local-address/*. This part of the HREF is the same for all the links that ODS creates.
- The value of the BODY= option, **`grain-body`**, provides the basis for the next part of the HREF. However, because the NEWFILE= option creates a new file for each output object, ODS increments this base value each time that it creates a file. The resulting filenames become part of the HREF. They are Grain-Body.htm, Grain-Body1.htm, Grain-Body2.htm, and Grain-Body3.htm.
- The value of the ANCHOR= option provides the basis for the last part of the HREF, which follows the pound sign (#). Because the ANCHOR= option is not used in this example, ODS uses the default value of IDX. With each use, ODS increments the value of the anchor.

Example 2: Appending to HTML Files

ODS features:
- ODS HTML statement:
  - Argument:
    - BODY= with a fileref
      - NO_BOTTOM_MATTER suboption
      - NO_TOP_MATTER suboption
  - Options:
    - ANCHOR=
    - STYLE=

Other SAS features:
- FILENAME statement
- PROC PRINT
- PROC REPORT
- DATA _NULL_ statement

Data set:
See “Creating the Grain_Production Data Set” on page 878.

Format:
See “Creating the $CNTRY Format” on page 869.

Program Description The following example creates HTML output from PROC PRINT and PROC REPORT. It also uses the DATA step to write customized HTML code to the file that contains the HTML output. The DATA step executes between procedure steps.

Program

Close the LISTING destination so that no listing output is produced. The ODS LISTING statement closes the LISTING destination to conserve resources. If the destination is left open, then ODS will produce both Listing and HTML output.

```
ods listing close;
options obs=10;
```

Assign a fileref to the file GrainReport.html. The FILENAME statement assigns the fileref REPORTS to the file GrainReport.html that will contain the HTML output.

```
filename reports 'GrainReport.html';
```

Create HTML output and suppress the writing of the default HTML code that would be written at the end of the file. The ODS HTML statement opens the HTML destination and creates HTML output. The NO_BOTTOM_MATTER option suppresses the writing of the default HTML code that, by default, ODS writes at the end of a file.

```
ods html body=reports (no_bottom_matter)
```

Specify the style definition for formatting the HTML output. The STYLE= option specifies that the style D3D be used.

```
          style=D3D;
```

Create a report that contains only the data from 1996. Select and format the variables that you want to include, specify a title, and specify a footnote. This PROC PRINT step prints the observations in the data set Grain_Production that have a value of 1996 for the variable Year. The VAR statement selects Country, Type, and Kilotons as the variables that you want to be displayed in the output. The TITLE and FOOTNOTE statements specify the title and footnote.

```
proc print data=grain_production;
   var country type kilotons;
   format country $cntry. kilotons comma12.;
   where year=1996;
   title 'Leading Grain-Producing Countries';
   footnote 'Measurements are in metric tons.';
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it.

```
ods html close;
```

Assign the fileref REPORTS to the file 'GrainReport.html'. This FILENAME statement assigns a fileref to the file to be updated, GrainReport.html. The MOD option opens the file in update mode.

Operating Environment Information: The MOD option might not be valid in all operating environments. See your operating environment documentation for more information. △

```
filename reports 'GrainReport.html' mod;
```

Append text to the HTML file REPORTS. This DATA step writes to the file that is referenced by REPORTS. The PUT statements create an H2 header in the HTML file.

```
data _null_;
   file reports;
   put ''<h2>The preceding output is from PROC PRINT.'';
   put ''I am going to try a variety of procedures.'';
   put ''Let me know which procedure you prefer.'';
   put ''By the way, this report uses the D3D style.</h2>'';
run;
```

Create HTML output. This ODS HTML statement opens the HTML destination and creates HTML output. The NO_TOP_MATTER and the NO_BOTTOM_MATTER suboptions suppress the default HTML code that ODS writes to the top and the bottom of a file.

```
ods html body=reports (no_top_matter no_bottom_matter)
```

Specify the root name for the HTML anchor tags. The ANCHOR= option specifies **report** as the root name for the HTML anchor tags.

Note: When you use ODS to append to an HTML file that ODS created, you must specify a new anchor name each time that you open the file from ODS so that you do not write the same anchors to the file again. (ODS cannot recognize anchors that are already in the file when it opens it, and by default it uses **IDX** as the base for anchor names). △

```
          anchor='report';
```

Create a report that contains only the 1996 data. The PROC REPORT step prints the data set. ODS adds HTML output to the body file. The NOWINDOWS option specifies that PROC REPORT runs without the REPORT window and sends its output to the open output destination(s).

```
proc report data=grain_production nowindows;
   where year=1996;
   column country type kilotons;
   define country / group width=14 format=$cntry.;
   define type / group 'Type of Grain';
   define kilotons / format=comma12.;
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it.

```
ods html close;
```

Append text to the HTML file REPORTS. This DATA step writes to the file that is referenced by REPORTS. The PUT statements create an H2 header in the HTML file.

```
data _null_;
   file reports;
   put ''<h2>The preceding output is from PROC REPORT.'';
   put ''It doesn't repeat the name of the country on every line.'';
   put ''This report uses the default style.</h2>'';
run;
```

Create HTML output to write the bottom matter to the file, repress the printing of the top matter, and provide a new root name for the anchor tags. In order to write the bottom matter to the HTML file so that it contains valid HTML code, you must open the HTML destination one more time. NO_TOP_MATTER ensures that the top matter is not placed in the file again. ANCHOR= provides a new root name for the anchors in the bottom matter.

```
ods html body=reports(no_top_matter)anchor='end';
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are open for that destination.

```
ods html close;
```

HTML Output

Display 5.6 HTML Output with Appended HTML

This output is created by appending HTML output to an existing HTML file.

Leading Grain-Producing Countries

Obs	Country	Type	Kilotons
16	Brazil	Wheat	3,302
17	Brazil	Rice	10,035
18	Brazil	Corn	31,975
19	China	Wheat	109,000
20	China	Rice	190,100
21	China	Corn	119,350
22	India	Wheat	62,620
23	India	Rice	120,012
24	India	Corn	8,660
25	Indonesia	Wheat	.

Measurements are in metric tons.

The preceding output is from PROC PRINT. I am going to try a variety of procedures. Let me know which procedure you prefer. By the way, this report uses the D3D style.

Leading Grain-Producing Countries

Country	Type of Grain	Kilotons
Brazil	Corn	31,975
	Rice	10,035
	Wheat	3,302
China	Corn	119,350
	Rice	190,100
	Wheat	109,000
India	Corn	8,660
	Rice	120,012
	Wheat	62,620
Indonesia	Wheat	.

Measurements are in metric tons.

The preceding output is from PROC REPORT. It doesn't repeat the name of the country on every line. This report uses the default style.

See Also

Statements:

"ODS MARKUP Statement" on page 147

Appendix 2, "ODS and the HTML Destination," on page 891

ODS HTMLCSS Statement

Opens, manages, or closes the HTMLCSS destination, which produces HTML output with cascading style sheets.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS HTMLCSS< (<ID=>*identifier*)> <*action*>;

ODS HTMLCSS <(<ID=>*identifier*)><*option(s)*>;

Without an Action or Options

If you use the ODS HTMLCSS statement without an action or options, then it opens the HTMLCSS destination and creates HTMLCSS output.

Actions

The following table lists the actions available for the ODS HTMLCSS statement. The ODS HTMLCSS statement is part of the markup family of statements. For complete descriptions of these options, see"Actions" on page 147 in the ODS MARKUP statement.

Table 5.13 ODS HTMLCSS Action Summary Table

Task	Action
Close the HTMLCSS destination and the file that is associated with it	CLOSE
Exclude output objects from the HTMLCSS destination	EXCLUDE
Select output objects for the HTMLCSS destination	SELECT
Write to the SAS log the current selection or exclusion list for the HTMLCSS destination	SHOW

Options

The following table lists the options that are available for the ODS HTMLCSS statement, which is part of the markup family of statements. For complete descriptions of these options, see "Options" on page 148 in the ODS MARKUP statement.

Table 5.14 ODS HTMLCSS Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the HTMLCSS destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the HTMLCSS destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the HTMLCSS files that the destination writes to	METATEXT=

Task	Option
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the HTMLCSS destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the HTMLCSS destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS HTMLCSS statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

ODS HTML3 Statement

Opens, manages, or closes the HTML3 destination, which produces HTML 3.2 formatted output.

Valid: anywhere
Category: ODS: Third-Party Formatted

Syntax

ODS HTML3<(<ID=>*identifier*)> <*action*>;

ODS HTML3 <(<ID=>*identifier*)><*option(s)*>;

Without an Action or Options

If you use the ODS HTML3 statement without an action or options, then it opens the HTML3 destination and creates HTML3 output.

Actions

The following table lists the actions available for the ODS HTML3 statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.15 ODS HTML3 Action Summary Table

Task	Action
Close the HTML3 destination and the file that is associated with it	CLOSE
Exclude output objects from the HTML3 destination	EXCLUDE
Select output objects for the HTML3 destination	SELECT
Write to the SAS log the current selection or exclusion list for the HTML3 destination	SHOW

Options

The following table lists the options available for the ODS HTML3 statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.16 ODS HTML3 Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the HTML3 destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=

Task	Option
Open the HTML3 destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify a device for the HTML output destination	DEVICE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the HTML3 files that the destination writes to	METATEXT=
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the HTML3 destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the HTML3 destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=

Task	Option
Insert text into your document	TEXT=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS HTML3 statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

By default, the SAS registry is configured to generate HTML 4 output when you specify the ODS HTML statement. To permanently change the default HTML version to 3.2, you can change the setting of the HTML version in the SAS registry. The ODS HTML statement will then produce HTML 3.2 output. For information about how to change your default HTML version, see "Changing Your Default HTML Version Setting" on page 32.

See Also

Statements:
"ODS MARKUP Statement" on page 147
"ODS HTML Statement" on page 124
Appendix 2, "ODS and the HTML Destination," on page 891
"Changing SAS Registry Settings for ODS" on page 32

ODS IMODE Statement

Opens, manages, or closes the IMODE destination, which produces HTML output as a column of output, separated by lines.

Valid: anywhere
Category: ODS: Third-Party Formatted

Syntax

ODS IMODE < (<ID=>*identifier*)> <*action*>;

ODS IMODE (<ID=>*identifier*) <*option(s)*>;

Without an Action or Options

If you use the ODS IMODE statement without an action or options, then it opens the IMODE destination and creates IMODE output.

Actions

The following table lists the actions available for the ODS IMODE statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.17 ODS IMODE Action Summary Table

Task	Action
Close the IMODE destination and the file that is associated with it	CLOSE
Exclude output objects from the IMODE destination	EXCLUDE
Select output objects for the IMODE destination	SELECT
Write to the SAS log the current selection or exclusion list for the IMODE destination	SHOW

Options

The following table lists the options available for the ODS IMODE statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.18 ODS IMODE Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the IMODE destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the IMODE destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=

Task	Option
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the IMODE files that the destination writes to	METATEXT=
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the IMODE destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the IMODE destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=

Task	Option
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS IMODE statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

ODS LISTING Statement

Opens, manages, or closes the LISTING destination.

Valid: anywhere

Category: ODS: SAS Formatted

Syntax

ODS LISTING <*action*>;

ODS LISTING <DATAPANEL=*number* | DATA | PAGE > <FILE=*file-specification*>;

Without an Action or Options

If you use the ODS LISTING statement without an action or options, it opens the LISTING destination.

Actions

An *action* performs one of the following procedures:

- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

An *action* is one of the following:

CLOSE
closes the LISTING destination and any files that are associated with it.

Tip: When you close an ODS destination, ODS does not send output to that destination. Closing an unneeded destination frees some system resources.

EXCLUDE *exclusion(s)*| ALL | NONE
excludes one or more output objects from the LISTING destination.

Default: NONE

Restriction: The LISTING destination must be open for this action to take effect.

Main discussion: "ODS EXCLUDE Statement" on page 110

SELECT *selection(s)* | ALL | NONE
selects output objects for the LISTING destination.

Default: ALL

Restriction: The LISTING destination must be open for this action to take effect.

Main discussion: "ODS SELECT Statement" on page 264

SHOW
writes the current selection or exclusion list for the LISTING destination to the SAS log.

Restriction: The LISTING destination must be open for this action to take effect.

Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.

See also: "ODS SHOW Statement" on page 277

Options

DATAPANEL=*number* | DATA | PAGE
suggests how to split a table that is too wide to fit on a single page into sections of columns and rows. Each section of columns and rows is a *data panel*. Each data panel has column headings at the top.

Note: In this context, a page is what the procedure uses as a page in creating the listing output. The SAS system options LINESIZE= and PAGESIZE= generally determine the page size, although some procedures (PROC REPORT, for example) can temporarily override the values that the system options specify. △

number
writes the specified number of observations in a panel, if possible. More than one panel can occur on every page if space permits.

Range: 1 to the largest integer that the operating system supports

DATA
bases the size of the panel on the way that the table is stored in memory. This value provides the fastest performance. However, if the table contains many columns, the number of rows in each panel might be small.

PAGE
tries to make panels that match the page size. If the table contains more columns than can fit on a page, the first page is filled with as many observations as possible for as many columns as can fit on a single line. The second page contains the same observations for the next group of columns, and so on, until all rows and columns have been printed.

This arrangement minimizes the amount of space that is used for column headings because most pages contain observations for only one set of columns.

Restriction: If the page size is greater than 200, ODS uses DATAPANEL=200.

Default: PAGE

DEVICE= *device-driver*
specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

The following table lists default devices for the most common ODS output destinations.

Output Destination	Default Device
HTML	PNG
LISTING	Host Specific Display Device (PC- WIN, UNIX - XColor, MVS -Display Device)
Measured RTF	PNG
RTF	SASEMF
PCL	SASPRTM (Monochrome Output)*
PDF	SASPRTC (Color Output)*
POSTSCRIPT	SASPRTC (Color Output)*
PRINTER	Host Specific Default Printer*

* Does not support changing the default device in the SAS Registry.

Tip: Specifying a device on the ODS DEVICE= option takes precedence over the SAS global option and the graphics option.

See: "DEVICE= System Option" in *SAS Language Reference: Dictionary*. See "Device Drivers" in *SAS/GRAPH: Reference* for information on selecting device drivers.

FILE=*file-specification*
specifies the file to write to. *file-specification* is one of the following:

'external-file'
is the name of an external file to which to write.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref. (For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.)

Default: If you do not specify a file to write to, ODS writes the output to the LISTING window.

GPATH= *file-specification* <(url='*Uniform-Resource-Locator*' | NONE)>
specifies the location for all graphics output that is generated while the destination is open.

file-specification
specifies the file or SAS catalog to which to write. ODS names automatically each output object that it places in the file. If you specify an invalid filename, the ActiveX and Java devices send output to the default filename. Other devices create the file as a directory and write output to that directory using the default filename. For more information about how ODS names catalog entries and external files, see *SAS/GRAPH: Reference*. *file-specification* is one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref. For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

Interaction: If you specify a fileref in the GPATH= option, ODS does not use information from the GPATH= option when it constructs links.

libref.catalog
specifies a SAS catalog to which to write.

URL= '*Uniform-Resource-Locator*' | NONE
specifies a URL for *file-specification*.

Uniform-Resource-Locator
is the URL you specify. ODS uses this URL instead of the filename in all the links and references that it creates to the file.

Requirement: You must enclose *Uniform-Resource-Locator* in quotation marks.

NONE
specifies that no information from the GPATH= option appears in the links or references.

Tip: This option is useful for building output files that can be moved from one location to another. If the links from the contents and page files are constructed with a simple URL (one name), they will resolve if the contents, the page, and body files are all in the same location.

IMAGE_DPI=
specifies the image resolution of ODS graphics output. Output from device-based graphics is not affected.

Default: 100

Restriction: The IMAGE_DPI= option affects template-based graphics only.

PACKAGE *<package-name>*
specifies that the output from the destination be added to a package.

package-name
specifies the name of a package that was created with the ODS PACKAGE statement. If no name is specified, then the output is added to the unnamed package that was opened last.

See also:
"ODS PACKAGE Statement" on page 198

SGE= ON | OFF
determines whether you can edit ODS graphics output with the ODS Graphics Editor.

Default: OFF

Restriction: The SGE= option affects template-based graphics only.

See also: *SAS/GRAPH: ODS Graphics Editor User's Guide*

ODS MARKUP Statement

Opens, manages, or closes the MARKUP destination, which produces SAS output that is formatted using one of many different markup languages.

Valid: anywhere

Category: ODS: Third-Party Formatted

Interaction: The output type is determined by the TAGSET | TYPE= option, which specifies the kind of markup language that is applied to the output.

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
```

Syntax

ODS MARKUP <(<ID=>*identifier*)> <*action*>;

ODS MARKUP <(<ID=>*identifier*)> <*option(s)*><TAGSET=*tagset-name*> <*action*>;

Actions

An *action* is one of the following:

CLOSE
closes the destination and any files that are associated with it.

Tip: When an ODS destination is closed, ODS does not send output to that destination. Closing an unneeded destination conserves system resources.

Feature in: All examples

EXCLUDE *exclusion(s)* | ALL | NONE
excludes one or more output objects from the destination.

Default: NONE

Restriction: A destination must be open for this action to take effect.

Main discussion: “ODS EXCLUDE Statement” on page 110

SELECT *selection(s)* | ALL | NONE
selects output objects for the specified destination.

Default: ALL

Restriction: A destination must be open for this action to take effect.

Main discussion: “ODS SELECT Statement” on page 264

SHOW
writes the current selection or exclusion list for the destination to the SAS log.

Restriction: A destination must be open for this action to take effect.

See also: "ODS SHOW Statement" on page 277

Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list. For information about selection and exclusion lists, see "Selection and Exclusion Lists" on page 34.

Options

Table 5.19 ODS MARKUP Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view ODS HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the HTML destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the HTML destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a device for the output destination	DEVICE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=

Task	Option
Open multiple instances of the same destination at the same time	ID=
Specify the image resolution for graphical output	IMAGE_DPI =
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the HTML files that the destination writes to	METATEXT=
Create a new body file at the specified starting point	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the HTML destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the HTML destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Specify a keyword value for a tagset. A tagset is a template that defines how to create a markup language output type from a SAS format.	TAGSET=
Insert text into your document	TEXT=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

ANCHOR= '*anchor-name*'
specifies a unique base name for the anchor tag that identifies each output object in the current body file.

Each output object has an anchor tag for the contents, page, and frame files to reference. The links and references, which are automatically created by ODS, point to the name of an anchor. Therefore, each anchor name in a file must be unique.

anchor-name
is the base name for the anchor tag that identifies each output object in the current body file.

ODS creates unique anchor names by incrementing the name that you specify. For example, if you specify ANCHOR= 'tabulate', then ODS names the first anchor

tabulate. The second anchor is named **tabulate1**; the third is named **tabulate2**, and so on.

Requirement: You must enclose *anchor-name* in quotation marks.

Tip: You can change anchor names as often as you want by specifying the ANCHOR= option in a markup family statement anywhere in your program. Once you have specified an anchor name, it remains in effect until you specify a new one.

Restriction: Each anchor name in a file must be unique.

Interaction: If you open a file to append to it, then be sure to specify a new anchor name so that you do not write the same anchors to the file again. ODS does not recognize anchors that are already in a file when it opens the file.

Tip: Specifying new anchor names at various points in your program is useful when you want other web pages to link to specific parts of your markup language output. Because you can control where the anchor name changes, you know in advance what the anchor name will be at those points.

ARCHIVE='*string*'

The ARCHIVE= option is valid only for the GOPTIONS java device. The ARCHIVE= option enables you to specify which applet to use in order to view the ODS HTML output.

The string must be one that the browser can interpret. For example, if the archive file is local to the computer that you are running SAS on, you can use the FILE protocol to identify the file. If you want to point to an archive file that is on a web server, use the HTTP protocol.

Default: If you do not specify ARCHIVE= and you are using the JAVA device driver, ODS uses the value of the SAS system option APPLETOC=. This value points to the location of the Java archive files that ship with the SAS system. To find out what the value of this option is, you can either look in the Options window in the Files folder under Environment Control, or you can submit the following procedure step:

```
proc options option=appletloc;
run;
```

There is no default if you are using the ACTIVEX device driver.

Requirement: You must enclose *string* in quotation marks.

Requirement: The ARCHIVE attribute is a feature of Java 1.1. Therefore, if you are using the Java device driver, your browser must support this version of Java. Both Internet Explorer 4.01 and Netscape 4.05 support Java 1.1.

Interaction: Use ARCHIVE= in conjunction with SAS/GRAPH procedures and the DEVICE=JAVA or DEVICE=ACTIVEX option in the GOPTIONS statement.

Tip: Typically, this option should not be used, because the SAS server automatically determines the correct SAS/GRAPH applets to view the ODS HTML output. However, if you have renamed the JAR files, or have other applets with which to view the ODS HTML output, this option enables you to access these applets.

Tip: Use the CODEBASE= option to specify the file path. It is recommended that you do not put a file path in your ARCHIVE= option.

ATTRIBUTES= (*attribute-pair-1 ... attribute-pair-n*)

writes the specified attributes between the tags that generate dynamic graphics output.

attribute-pair
specifies the name and value of each attribute. *attribute-pair* has the following form:

'*attribute-name*'= '*attribute-value*'

attribute-name
is the name of the attribute.

attribute-value
is the value of the attribute.

Requirement: You must enclose *attribute-name* and *attribute-value* in quotation marks.

Interaction: Use the ATTRIBUTES= option in conjunction with SAS/GRAPH procedures and with the DEVICE=JAVA, JAVAMETA, or ACTIVEX options in the GOPTIONS statement.

See also: *SAS/GRAPH: Reference* for valid attributes for the following applets:

- Graph Applet
- Map Applet
- Contour Applet
- MetaView Applet

BASE= '*base-text*'
Specifies the text to use as the first part of all links and references that ODS creates in the output files.

base-text
is the text that ODS uses as the first part of all links and references that ODS creates in the file.
Consider this specification:

```
BASE= 'http://www.your-company.com/local-url/'
```

In this case, ODS creates links that begin with the string **http://www.*your-company.com/local-url/***. The appropriate *anchor-name* completes the link.

Requirement: You must enclose *base-text* in quotation marks.

BODY= '*file-specification*' <(*suboption(s)*)>
opens a markup family destination and specifies the file that contains the primary output that is created by the ODS statement. These files remain open until you do one of the following:

- close the destination with either an ODS *markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

file-specification
specifies the file, fileref, or SAS catalog to write to.
file-specification is one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

Restriction: The BODY=*fileref* option cannot be used in conjunction with the NEWFILE= option.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

entry.markup
specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= option on page 162.

(*suboption(s)*)
specifies one or more suboptions in parentheses. Suboptions are instructions for writing the output files. For a list of suboptions, see "Suboptions" on page 166.

Note: For some values of TAGSET=, this output will be an HTML file, for other TAGSET= values, the output will be an XML file, and so on. △

Alias: FILE=

Interaction: If you use the BODY= option in an ODS markup family statement that refers to an open ODS markup destination, the option will force ODS to close the destination and all files associated with it, and then to open a new instance of the destination. For more information see "Opening and Closing the MARKUP Destination" on page 167.

Featured in: All examples

CHARSET= *character-set*
specifies the character set to be generated in the META declaration for the HTML output.

See: For information about the CHARSET option, see *SAS National Language Support (NLS): Reference Guide*.

CODE= *'file-specification'* <(*suboption(s)*)>
opens a markup family destination and specifies the file that contains relevant style information, such as XSL (Extensible Stylesheet Language). These files remain open until you do one of the following:

- close the destination with either an ODS *markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

file-specification
specifies the file, fileref, or SAS catalog to write to.
file-specification is one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

entry.markup
specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= option.

suboption(s)
specifies one or more suboptions in parentheses. Suboptions are instructions for writing the output files. For a list of suboptions, see "Suboptions" on page 166.

CODEBASE='*string*'
specifies the location of the executable Java applet or the ActiveX control file. *string* is specified as a pathname or as a URL. The CODEBASE file path option has two definitions, depending on the GOPTIONS device used.

When you generate Web presentations with the JAVA and ActiveX device drivers, SAS generates HTML pages that automatically look for the JAVA archive files or the ActiveX control file in the default installation location.

For the ActiveX device:
If you use the ActiveX device driver with ODS to generate output containing an ActiveX control, then specify the CODEBASE= option in the ODS statement. The value of the CODEBASE= option should include the location and the version of the EXE file.

Tip: You do not need to specify the CODEBASE= option with the DEVICE=ACTIVEX option unless the users that view your output do not have the ActiveX control installed on their machine. When users that do not have the control installed view your output, they are prompted to download the control.

See also: *SAS/GRAPH: Reference* for information on specifying the location of control and applet files using the CODEBASE= and ARCHIVE= options.

For the Java device:
If you use the Java device driver with ODS to generate output containing a SAS/GRAPH applet, specify the path to the JAR file with the CODEBASE= option in the ODS statement.

When you specify DEVICE=JAVA, the users that view your output must have access to the appropriate Java applet. By default, SAS sets the value of CODEBASE= to refer to the executable file for the applet that is automatically installed with SAS. The default location of the SAS Java archive files is specified by the APPLETLOC= system option. If the default location is accessible by users who will be viewing your Web presentation, and the SAS Java archive is installed at that location, then you do not need to specify the CODEBASE= option.

Tip: Specify only the directory of the JAR file. The CODEBASE= location can be specified as a pathname or as a URL

See also: *SAS/GRAPH: Reference* for information on specifying the location of control and applet files using the CODEBASE= and ARCHIVE= options.

CONTENTS= '*file-specification*' <(*suboption(s)*)>
opens a markup family destination and specifies the file that contains a table of contents for the output. These files remain open until you do one of the following:

- close the destination with either an ODS *markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

file-specification
specifies the file, fileref, or SAS catalog to write to.
file-specification is one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

entry.markup
specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= option.

suboption(s)
specifies one or more suboptions in parentheses. Suboptions are instructions for writing the output files. For a list of suboptions, see "Suboptions" on page 166.

CSSSTYLE= *'file-specification'*<(*media-type-1 <...media-type-10>*)>
specifies a cascading style sheet to apply to your output.

file-specification
specifies a file, fileref, or URL that contains CSS code..
file-specification is one of the following:

"*external-file*"
is the name of the external file.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

"*URL*"
is a URL to an external file.

Requirement: You must enclose *external-file* in quotation marks.

(*media-type-1*<.. *media-type-10*>)
specifies one or more media blocks that corresponds to the type of media that your output will be rendered on. CSS uses media type blocks to specify how a document is to be presented on different media: on the screen, on paper, with a speech synthesizer, with a braille device, and so on.

The media block is added to your output in addition to the CSS code that is not contained in any media blocks. By using the *media-type* suboption, in addition to the general CSS code, you can import the section of a CSS file intended only for a specific media type.

Default: If no *media-type* is specified in your ODS statement, but you do have media types specified in your CSS file, then ODS uses the Screen media type.

Range: You can specify up to ten different media types.

Requirement: You must enclose *media-type* in parentheses.

Requirement: You must specify *media-type* next to the *file-specification* specified by the CSSSTYLE= option.

Tip: If you specify multiple media types, all of the style information in all of the media types is applied to your output. However, if there is duplicate style

information in different media blocks, then the styles from the last media block are used.

Interaction: If both the STYLE= option and the CSSSTYLE= option are specified on an ODS statement, the option specified last it the option that is used.

Requirement: CSS files must be written in the same type of CSS produced by the ODS HTML statement. Only class names are supported, with no IDs and no context based selectors. To view the CSS code that ODS creates, you can specify the STYLESHEET= option, or you can view the source of an HTML file and look at the code between the <STYLE> </STYLE> tags at the top of the file. For an example of a valid for ODS CSS file, see Example 6 on page 178.

Featured in: Example 6 on page 178

DEVICE= *device-driver*

specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

The following table lists default devices for the most common ODS output destinations.

Output Destination	Default Device
HTML	PNG
LISTING	Host Specific Display Device (PC- WIN, UNIX - XColor, MVS -Display Device)
Measured RTF	PNG
RTF	SASEMF
PCL	SASPRTM (Monochrome Output)*
PDF	SASPRTC (Color Output)*
POSTSCRIPT	SASPRTC (Color Output)*
PRINTER	Host Specific Default Printer*

* Does not support changing the default device in the SAS Registry.

Tip: Specifying a device on the ODS DEVICE= option takes precedence over the SAS global option and the graphics option.

See: "Device= System Option" in *SAS Language Reference: Dictionary*. See "Device Drivers" in *SAS/GRAPH: Reference* for information on selecting device drivers.

ENCODING= *local-character-set-encoding*

overrides the encoding for input or output processing (transcodes) of external files.

See: For information about the ENCODING= option, see *SAS National Language Support (NLS): Reference Guide*.

EVENT=*event-name* (FILE= | FINISH | LABEL= | NAME= | START | STYLE= |TARGET= | TEXT= | URL=)

specifies an event and the value for event variables that are associated with the event.

(FILE= BODY | CODE | CONTENTS | DATA | FRAME | PAGES | STYLESHEET);

triggers one of the known types of output files that correspond to the BODY=, CODE=, CONTENTS=, FRAME=, PAGES=, and STYLESHEET= options.

(FINISH)
triggers the finish section of an event.

See: For information about events, see "Understanding Events" on page 839.

(LABEL='*variable-value*')
specifies the value for the LABEL event variable.

Requirement: *variable-value* must be enclosed in quotation marks.

See: For information about the LABEL event variable, see "Event Variables" on page 833.

(NAME='*variable-value*')
specifies the value for the NAME event variable.

Requirement: *variable-value* must be enclosed in quotation marks.

See: For information about the NAME event variable, see "Event Variables" on page 833.

(START)
triggers the start section of an event.

See: For information about events, see "Understanding Events" on page 839.

(STYLE=*style-element*)
specifies a style element.

See: For information about style elements, see "Style Attributes and Their Values" on page 498.

(TARGET='*variable-value*')
specifies the value for the TARGET event variable.

Requirement: *variable-value* must be enclosed in quotation marks.

See: For information about the TARGET event variable, see "Event Variables" on page 833.

(TEXT='*variable-value*')
specifies the value for the TEXT event variable.

Requirement: *variable-value* must be enclosed in quotation marks.

See: For information about the TEXT event variable, see "Event Variables" on page 833.

(URL='*variable-value*')
specifies the value for the URL event variable.

Requirement: *variable-value* must be enclosed in quotation marks.

See: For information about the URL event variable, see "Event Variables" on page 833.

Default: (FILE='BODY')

Requirement: The EVENT= option's suboptions must be enclosed in parenthesis.

FRAME= *'file-specification'* <(*suboption(s)*)>
opens a markup family destination and, for HTML output, specifies the file that integrates the table of contents, the page contents, and the body file. If you open the frame file, then you see a table of contents, a table of pages, or both, as well as the body file. For XLM output, FRAME= specifies the file that contains the DTD. These files remain open until you do one of the following:

- close the destination with either an ODS *markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

file-specification
specifies the file, fileref, or SAS catalog to write to.
file-specification is one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

entry.markup
specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= option.

suboption(s)
specifies one or more suboptions in parentheses. Suboptions are instructions for writing the output files. For a list of suboptions, see "Suboptions" on page 166.

Restriction: If you specify the FRAME= option, then you must also specify the CONTENTS= option, the PAGE= option, or both.

Featured in: Example 2 on page 172

GFOOTNOTE | NOGFOOTNOTE
controls the location where footnotes are printed in the graphics output.

GFOOTNOTE
prints footnotes that are created by SAS/GRAPH, the SGPLOT procedure, the SGPANEL procedure, or the SGSCATTER procedure. The footnotes appear inside the graph borders.

NOGFOOTNOTE
prints footnotes that are created by ODS, which appears outside the graph borders.

Default: GFOOTNOTE

Restriction: Footnotes that are displayed by a markup language statement support all SAS/GRAPH FOOTNOTE statement options. The font must be valid for the browser. Options that ODS cannot handle, such as text angle specifications, are ignored. For details about the SAS/GRAPH FOOTNOTE statement, see *SAS/GRAPH: Reference*.

Restriction: This option applies only to SAS programs that produce one or more device-based graphics, or graphics created by the SGPLOT procedure, the SGPANEL procedure, or the SGSCATTER procedure.

GPATH= *'aggregate-file-storage-specification'* | *fileref* | *libref.catalog* (URL= *'Uniform-Resource-Locator'* | NONE)
specifies the location for all graphics output that is generated while the destination is open. Use this option when you want to write graphics output files to a location different that specified by the PATH= option for markup files. If you specify an invalid filename, the ActiveX and Java devices send output to the default filename. Other devices create the file as a directory and write output to that directory using the default filename. For more information about how ODS names catalog entries and external files, see *SAS/GRAPH: Reference*.

'*aggregate-file-storage-location*'
specifies an aggregate storage location such as directory, folder, or partitioned data set.

Requirement: You must enclose *aggregate-file-storage-location* in quotation marks.

fileref
is a file reference that has been assigned to an aggregate storage location. Use the FILENAME statement to assign a fileref. For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

Interaction: If you specify a fileref in the GPATH= option, then ODS does not use information from the GPATH= option when it constructs links.

libref.catalog
specifies a SAS catalog to write to.

URL= '*Uniform-Resource-Locator*' | NONE
specifies a URL for *file-specification*.

Uniform-Resource-Locator
is the URL you specify. ODS uses this URL instead of the filename in all the links and references that it creates to the file.

Requirement: You must enclose *Uniform-Resource-Locator* in quotation marks.

NONE
specifies that no information from the GPATH= option appears in the links or references.

Tip: This option is useful for building output files that can be moved from one location to another. If the links from the contents and page files are constructed with a simple URL (one name), then they will resolve, as long as the contents, page, and body files are all in the same location.

Default: If you omit the GPATH= option, then ODS stores graphics in the location that is specified by the PATH= option. If you do not specify the PATH= option, then ODS stores the graphics in the current directory. For more information, see the PATH= option on page 162.

GTITLE | NOGTITLE
controls the location where titles are printed in the graphics output.

GTITLE
prints the title that is created by SAS/GRAPH, the SGPLOT procedure, the SGPANEL procedure, or the SGSCATTER procedure. The title appears inside the graph borders

NOGTITLE
prints the title that is created by ODS, which appears outside of the graph borders.

Default: GTITLE

Restriction: Titles that are displayed by any markup language statement support most SAS/GRAPH TITLE statement options. The font must be valid for the browser. Options that ODS cannot handle, such as text angle specifications, are ignored. For details about the SAS/GRAPH TITLE statement, see *SAS/GRAPH: Reference*.

Restriction: This option applies only to SAS programs that produce one or more device-based graphics, or graphics created by the SGPLOT procedure, the SGPANEL procedure, or the SGSCATTER procedure.

HEADTEXT= '*markup-document-head*'

specifies markup tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to.

markup-document-head
specifies the markup tags to place between the <HEAD> and </HEAD> tags.

Requirement: You must enclose *markup-document-head* in quotation marks.

Tip: ODS cannot parse the markup that you supply. It should be well-formed markup that is correct in the context of the <HEAD> and </HEAD> tags.

Tip: Use the HEADTEXT= option to define programs (such as JavaScript) that you can use later in the file.

(ID= *identifier*)
enables you to run multiple instances of the same destination at the same time. Each instance can have different options.

identifier
specifies another instance of the destination that is already open. *identifier* is numeric or a series of characters that begin with a letter or an underscore. Subsequent characters can include letters, underscores, and numeric characters.

Restriction: If *identifier* is numeric, it must be a positive integer.

Requirement: The ID= option must be specified immediately after the ODS *MARKUP*|*TAGSET* statement keywords.

Tip: You can omit the ID= option, and instead use a name or a number to identify the instance.

Featured in: Example 1 on page 212

IMAGE_DPI=
specifies the image resolution for graphical output.

Restriction: The IMAGE_DPI= option affects template-based graphics only.

METATEXT= '*metatext-for-document-head*'
specifies HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags of all the HTML files that the destination writes to.

'*metatext-for-document-head*'
specifies the HTML code that provides the browser with information about the document that it is loading. For example, this attribute could specify the content type and the character set to use.

Requirement: You must enclose *metatext-for-document-head* in quotation marks.

Default: If you do not specify METATEXT=, then ODS writes a simple <META> tag, which includes the content-type of the document and the character set to use, to all the HTML files that it creates.

Tip: ODS cannot parse the HTML code that you supply. It should be well-formed HTML code that is correct in the context of the <HEAD> tags. If you are using METATEXT= as it is intended, then your META tag should look like this:

```
<META your-metatext-is-here>
```

Restriction: METATEXT= cannot exceed 256 characters.

NEWFILE= *starting-point*
creates a new body file at the specified *starting-point*.

starting-point
is the location in the output where you want to create a new body file.

ODS automatically names new files by incrementing the name of the body file. In the following example, ODS names the first body file **REPORT.XML**. Additional body files are named **REPORT1.XML**, **REPORT2.XML**, and so on.

Example:

```
BODY= 'REPORT.XML'
```

starting-point is one of the following:

BYGROUP
starts a new file for the results of each BY group.

NONE
writes all output to the body file that is currently open.

OUTPUT
starts a new body file for each output object. For SAS/GRAPH this means that ODS creates a new file for each SAS/GRAPH output file that the program generates.

Alias: TABLE

PAGE
starts a new body file for each page of output. A page break occurs when a procedure explicitly starts a new page (not because the page size was exceeded) or when you start a new procedure.

PROC
starts a new body file each time that you start a new procedure.

Default: NONE

Restriction: The NEWFILE= option cannot be used in conjunction with the BODY=*fileref* option.

Tip: If you end the filename with a number, then ODS begins incrementing with that number. In the following example, ODS names the first body file **MAY5.XML**. Additional body files are named **MAY6.XML**, **MAY7.XML**, and so on.

Example:

```
BODY= 'MAY5.XML'
```

NOGFOOTNOTE

See: GFOOTNOTE | NOGFOOTNOTE options

NOGTITLE

See: GTITLE | NOGTITLE options

OPTIONS (DOC= | *<suboption(s)>*)
specifies tagset-specific suboptions and a named value.

(DOC='QUICK' | 'HELP' | 'SETTINGS')
provides information about the specified tagset.

QUICK
describes the options available for this tagset.

HELP
provides generic help and information with a quick reference.

SETTINGS

provides the current option settings.

Requirement: All values must be enclosed in quotation marks.

suboption(s)
specifies one or more suboptions that are valid for the specified tagset. Supoptions have the following format:

keyword='value'

You can get information about suboptions for a specific tagset by specifying one of the following options when opening an ODS tagset statement or at any time after the destination has been opened.

- `options(doc='help');`
- `options(doc='quick');`
- `options(doc='settings');`

Requirement: The OPTION suboption's must be enclosed in parentheses.

Featured in: Example 1 on page 285

PACKAGE <*package-name*>
specifies that the output from the destination be added to a package.

package-name
specifies the name of a package that was created with the ODS PACKAGE statement. If no name is specified, then the output is added to the unnamed package that was opened last.

See also: "ODS PACKAGE Statement" on page 198

Featured in: Example 1 on page 202

PAGE= '*file-specification*' <(*suboption(s)*)>
opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file. ODS produces a new page of output whenever a procedure requests a new page. These files remain open until you do one of the following:

- close the destination with either an ODS *markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

file-specification
specifies the file, fileref, or SAS catalog to write to.
file-specification is one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

entry.markup
specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= option.

suboption(s)
specifies one or more suboptions in parentheses. Suboptions are instructions for writing the output files. For a list of suboptions, see "Suboptions" on page 166.

Interaction: The SAS system option PAGESIZE= has no effect on pages in HTML output except when you are creating batch output. For information about the PAGESIZE= option see *SAS Language Reference: Dictionary*.

PARAMETERS= (*parameter-pair-1 ... parameter-pair-n*)
writes the specified parameters between the tags that generate dynamic graphics output.

parameter-pair
specifies the name and value of each parameter. *parameter-pair* has the following form:

'*parameter-name*'= '*parameter-value*'

parameter-name
is the name of the parameter.

parameter-value
is the value of the parameter.

Requirement: You must enclose *parameter-name* and *parameter-value* in quotation marks.

Interaction: Use PARAMETERS= in conjunction with SAS/GRAPH procedures and the DEVICE=JAVA, JAVAMETA, or ACTIVEX options in the GOPTIONS statement.

See also: *SAS/GRAPH: Reference* for valid parameters for the following applets:

- Graph Applet
- Map Applet
- Contour Applet
- MetaView Applet

PATH= *'aggregate-file-storage-specification'* | *fileref* | *libref.catalog* (URL= *'Uniform-Resource-Locator'* | NONE)
specifies the location of an aggregate storage location or a SAS catalog for all markup files. If the GPATH= option is not specified, all graphics output files are written to the *"aggregate-file-storage-specification"* or *libref*.

'*aggregate-file-storage-location*'
specifies an aggregate storage location such as directory, folder, or partitioned data set.

Requirement: You must enclose *aggregate-file-storage-location* in quotation marks.

fileref
is a file reference that has been assigned to an aggregate storage location. Use the FILENAME statement to assign a fileref.

Interaction: If you use a fileref in the PATH= option, then ODS does not use information from PATH= when it constructs links.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

libref.catalog
specifies a SAS catalog to write to.

See: For information about the LIBNAME statement, see *SAS Language Reference: Dictionary*.

URL= '*Uniform-Resource-Locator*' | NONE
specifies a URL for the *file-specification*.

Uniform-Resource-Locator
is the URL you specify. ODS uses this URL instead of the filename in all the links and references that it creates to the file.

NONE
specifies that no information from the PATH= option appears in the links or references.

Tip: This option is useful for building output files that can be moved from one location to another. The links from the contents and page files must be constructed with a single-name URL, and the contents, page, and body files must be in the same location.

Interaction: If you use the BODY= or FILE= external file option in conjunction with the PATH= option, the external file specification should not include path information.

RECORD_SEPARATOR= '*alternative-separator*' | NONE
specifies an alternative character or string that separates lines in the output files.

Different operating environments use different separator characters. If you do not specify a record separator, then the files are formatted for the environment where you run the SAS job. However, if you are generating files for viewing in a different operating environment that uses a different separator character, then you can specify a record separator that is appropriate for the target environment.

alternative-separator
represents one or more characters in hexadecimal or ASCII format. For example, the following option specifies a record separator for a carriage return character and a linefeed character for use with an ASCII file system:

```
RECORD_SEPARATOR= '0D0A'x
```

Operating Environment Information: In a mainframe environment, the option that specifies a record separator for a carriage return character and a linefeed character for use with an ASCII file system is:

```
RECORD_SEPARATOR= '0D25'x
```

△

Requirement: You must enclose *alternative-separator* in quotation marks.

NONE
produces the markup language that is appropriate for the environment where you run the SAS job.

Operating Environment Information: In a mainframe environment, by default, ODS produces a binary file that contains embedded record separator characters. This binary file is not restricted by the line-length restrictions on ASCII files. However, if you view the binary files in a text editor, then the lines run together.

If you want to format the files so that you can read them with a text editor, then use RECORD_SEPARATOR= NONE. In this case, ODS writes one line of markup language at a time to the file. When you use a value of NONE, the logical record length of the file that you are writing to must be at least as long as the longest line that ODS produces. If the logical record length of the file is not long enough, then the markup language might wrap to another line at an inappropriate place. △

Alias:

RECSEP=

RS=

STYLE= *style-definition*

specifies the style definition to use in writing the output files.

style-definition

describes how to display the presentation aspects (color, font face, font size, and so on) of your SAS output. A style definition determines the overall appearance of the documents that use it. Each style definition consists of style elements.

Main discussion: For a complete discussion of style definitions, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.

Interaction: The STYLE= option is not valid when you are creating XML output.

Default: If you do not specify a style definition, then ODS uses the file that is specified in the SAS registry subkey **ODS ▶ DESTINATIONS ▶ MARKUP**. By default, this value specifies **`Default`**.

Interaction: If you specify the STYLE= option on an ODS HTML4 statement and want to change the style definition with another ODS HTML4 statement , you must close the first statement before specifying the second statement, in order for any PROC PRINT output to use the second style definition.

STYLESHEET= *'file-specification'* <(*suboption(s)*)>

opens a markup family destination and places the style information for markup output into an external file, or reads style sheet information from an existing file. These files remain open until you do one of the following:

- close the destination with either an ODS *markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

file-specification

specifies the file, fileref, or SAS catalog to write to.
file-specification is one of the following:

external-file

is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref

is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

entry.markup

specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= option.

suboption(s)

specifies one or more suboptions in parentheses. Suboptions are instructions for writing the output files. For a list of suboptions, see "Suboptions" on page 166.

Note: By default, if you do not specifically send the information to a separate file, then the style sheet information is included in the specified HTML file. △

Featured in: Example 5 on page 176

TAGSET= *tagset-name*
specifies a keyword value for a tagset. A tagset is a template that defines how to create a markup language output type from a SAS format. Tagsets produce markup output such as Hypertext Markup Language (HTML), Extensible Markup Language (XML), and LaTeX.

An alternate form for specifying a tagset is as follows:

ODS *directory.tagset-name file-specification(s)<option(s)>*;

ODS *directory.tagset-name action*;

A *directory* can be TAGSET, a user defined entry, or a libref. By default, the tagsets that SAS supplies are located in the directory TAGSETS, which is within the item store SASUSER.TMPLMST. For more information about user defined tagsets and item stores, see Chapter 7, "TEMPLATE Procedure: Overview," on page 395.

Alias: TYPE=

Default: If you do not specify a TAGSET= value, then the ODS MARKUP statement defaults to XML output.

Interaction: If you use the TAGSET= option in an ODS markup family statement that refers to an open ODS markup destination, then the option will force ODS to close the destination and all files associated with it, and then to open a new instance of the destination. For more information, see "Opening and Closing the MARKUP Destination" on page 167.

Tip: SAS provides a set of tagset definitions. To get a list of the tagset names that SAS supplies, plus any tagsets that you created and stored in the SASUSER.TMPLMST template store, submit the following SAS statements:

```
proc template;
   list tagsets;
run;
```

See: For a list of valid tagsets and their descriptions, see "ODS Tagset Statement" on page 278.

See also: For additional information about specifying tagsets, see Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795.

Featured in: Example 2 on page 172, Example 3 on page 173, Example 4 on page 176

TEXT=*text-string*
inserts text into your document by triggering the paragraph event and specifying a text string to be assigned to the VALUE event variable.

Default: By default the TEXT= option is used in a paragraph event.

Tip: You can specify a *text-string* for a specific event by using the TEXT= option with the EVENT= option by using the following syntax:

EVENT=*event-name* (TEXT=*text-string*)

Featured in: Example 1 on page 114

See also: For information about events and event variables, see Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795.

TRANTAB= '*translation-table*'
specifies the translation table to use when transcoding a file for output.

See: For information about the TRANTAB= option, see *SAS National Language Support (NLS): Reference Guide*.

Suboptions

The following suboptions can be used with the BODY=, CODE=, CONTENTS=, FRAME=, PAGE=, and STYLESHEET= options:

(NO_BOTTOM_MATTER)
specifies that no ending markup language source code be added to the output file or.

Alias: NOBOT

Requirement: You must enclose NO_BOTTOM_MATTER in parentheses.

Requirement: You must specify NO_BOTTOM_MATTER next to the *file-specification* specified by the BODY=, CONTENTS=, PAGE=, FRAME=, or STYLESHEET= option, or next to the *tagset-name* specified by the TAGSET= option.

Requirement: If you append text to an external file you must use a FILENAME statement with the appropriate option for the operating environment.

Interaction: The NO_BOTTOM_MATTER suboption, in conjunction with the NO_TOP_MATTER suboption, makes it possible for you to add output to an existing file and then to put your own markup language between output objects in the file.

Interaction: When you are opening a file that ODS has previously written to, you must use the ANCHOR= option to specify a new base name for the anchors in order to avoid duplicate anchors.

Tip: If you want to leave a body file in a state that you can append to with ODS, then use NO_BOTTOM_MATTER with the *file-specification* in the BODY= option in any markup language statement.

See also: NO_TOP_MATTER on page 166

(NO_TOP_MATTER)
specifies that no beginning markup language source code be added to the top of the output file. For HTML 4.0, the NO_TOP_MATTER option removes the style sheet.

Alias: NOTOP

Requirement: You must enclose NO_TOP_MATTER in parentheses.

Requirement: You must specify NO_TOP_MATTER next to the *file-specification* specified by the BODY=, CONTENTS=, PAGE=, FRAME=, or STYLESHEET= option, or next to the *tagset-name* specified by the TAGSET= option.

Requirement: If you append text to an external file you must use a FILENAME statement with the appropriate option for the operating environment.

Interaction: The NO_TOP_MATTER suboption, in conjunction with the NO_BOTTOM_MATTER suboption, makes it possible for you to add output to an existing file and then to put your own markup language between output objects in the file.

Interaction: When you are opening a file that ODS has previously written to, you must use the ANCHOR= option to specify a new base name for the anchors in order to avoid duplicate anchors.

See also: NO_BOTTOM_MATTER on page 166 and ANCHOR= on page 149

(TITLE='*title-text*')
inserts into the metadata of a file, the text string that you specify as the text to appear in the browser window title bar.

title-text
is the text in the metadata of a file that indicates the title.

Requirement: You must enclose TITLE= in parentheses.

Requirement: You must enclose *title-text* in quotation marks.

Tip: If you are creating a web page that uses frames, then it is the TITLE= specification for the frame file that appears in the browser window title bar.

Featured in: Example 3 on page 173

(URL= '*Uniform-Resource-Locator*')
specifies a URL for the *file-specification*. ODS uses this URL (instead of the filename) in all the links and references that it creates and that point to the file.

Requirement: You must enclose URL= '*Uniform-Resource-Locator*' in parentheses.

Requirement: You must enclose *Uniform-Resource-Locator* in quotation marks.

Requirement: You must specify URL= '*Uniform-Resource-Locator*' next to the *file-specification* specified by the BODY=, CONTENTS=, PAGE=, FRAME=, or STYLESHEET= option, or next to the *tagset-name* specified by the TAGSET= option.

Tip: This option is useful for building HTML files that can be moved from one location to another. The links from the contents and page files must be constructed with a single name URL, and the contents, page, and body files must all be in the same location.

Tip: You never need to specify this suboption with the FRAME= option because ODS files do not reference the frame file.

Featured in: Example 5 on page 176

(DYNAMIC)
enables you to send output directly to a web server instead of writing it to a file. This option sets the value of the CONTENTTYPE= style attribute. For more information see the CONTENTTYPE= on page 516 style attribute in PROC TEMPLATE.

Default: If you do not specify DYNAMIC, then ODS sets the value of HTMLCONTENTTYPE= for writing to a file.

Requirement: You must enclose DYNAMIC in parentheses.

Requirement: You must specify DYNAMIC next to the *file-specification* specified by the BODY=, CONTENTS=, PAGE=, FRAME=, or STYLESHEET= option, or next to the *tagset-name* specified by the TAGSET= option.

Restriction: If you specify the DYNAMIC suboption with the BODY=, CONTENTS=, PAGE=, FRAME=, STYLESHEET= or TAGSET= option in the ODS HTML statement, then you must specify it for all the BODY=, CONTENTS=, PAGE=, FRAME=, STYLESHEET= or TAGSET= options in that statement.

Details

Opening and Closing the MARKUP Destination

You can modify an open MARKUP destination with many ODS MARKUP options. However, the BODY= and TAGSET= options will automatically close the open destination that is referred to in the ODS MARKUP statement, and will also close any files associated with it, and then will open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

Specifying Multiple ODS Destinations

The ODS MARKUP statement opens or closes one destination. Like all single output destinations, you can have only one markup destination open at one time, unless you use the ID= option.

However, you can specify multiple simultaneous ODS destinations to produce multiple markup output by doing both of the following:

- specifying some of the TAGSET= value keywords as a destination

- specifying any two-level tagset name, such as TAGSETS.PYX, TAGSETS.STYLE_DISPLAY, or one of your own tagset names.

Specifying a Tagset Keyword As an ODS Destination You can specify some tagset keywords as ODS destinations. The tagset determines the type of markup that you will have in your output file. For example, either of the following sets of statements are acceptable:

```
ods markup body='class.html' tagset=phtml;
...more SAS statements...
ods markup close;


ods phtml body='class.html';
...more SAS statements...
ods phtml close;
```

The ODS statement that you use to close a destination must be in the same form as the ODS statement that you used to open the destination. Therefore, the following is not acceptable, because SAS considers MARKUP and PHTML as separate destinations.

```
ods markup body='class.html' tagset=phtml;
...more SAS statements...
ods phtml close;
```

The tagsets that you can specify as both a TAGSET= value for ODS MARKUP or as a separate ODS destination are as follows:

CHTML
CSV
CSVALL
DOCBOOK
HTML4
HTMLCSS
IMODE
LATEX
PHTML
SASREPORT
TROFF
WML
WMLOLIST

Specifying a Two-Level Tagset Name As an ODS Destination You can open a destination by specifying the markup that you want to produce by naming its two-level tagset name. You can specify all tagsets in this manner. For example, the following ODS statements open the SASIOXML and MYTAGSET destinations. The ODS _ALL_ CLOSE statement closes the SASIOXML and MYTAGSET destinations as well as all other open destinations.

```
ods tagsets.sasioxml body='test1.xml';
ods tagsets.mytagset body='test2.xml';
...more SAS statements...
ods _all_ close;
```

You can also specify tagset names as follows, using the TYPE= option with a two-level tagset name:

```
ods markup type=tagsets.sasioxml body='test.xml';
```

Examples

Example 1: Creating an XML FILE

ODS features:
    ODS LISTING statement:
        Action:
            CLOSE
    ODS MARKUP statement:
        Action:
            CLOSE
        Options:
            BODY=

Other SAS features:
    PROC PRINT

Data Set:
See "Creating the StatePop Data Set" on page 881.

Program Description The following ODS MARKUP example creates XML markup from PRINT procedure output. The TAGSET= option for the ODS MARKUP statement is not specified, which defaults to XML output.

Program

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create XML output. The ODS MARKUP BODY= statement creates an XML file.

```
ods markup body='population.xml';
```

Print the data set. The PRINT procedure prints the data set StatePop.

```
proc print data=statepop;
run;
```

Close the MARKUP destination. The ODS MARKUP CLOSE statement closes the MARKUP destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```
ods markup close;
```

XML Output The following partial output is tagged with XML (Extensible Markup Language) tags.

Output 5.1 XML Markup from PRINT Procedure Output

```
<?xml version="1.0" encoding="windows-1252"?>

<odsxml>
<head>
<meta operator="user"/>
</head>
<body>
<proc name="Univariate">
<label name="IDX"/>
<title class="SystemTitle" toc-level="1">US Census of Population and Housing</title>
<proc-title class="ProcTitle" toc-level="1">The UNIVARIATE Procedure</proc-title>
<proc-title class="ProcTitle" toc-level="1">Variable:  CityPop_90  (1990 metropolitan pop in millions)</proc-title>
<branch name="Univariate" label="The Univariate Procedure" class="ContentProcName" toc-level="1">
<branch name="CityPop_90" label="CityPop_90" class="ContentFolder" toc-level="2">
<leaf name="Moments" label="Moments" class="ContentItem" toc-level="3">
<output name="Moments" label="Moments" clabel="Moments">
<output-object type="table" class="Table">

  <style>
    <border spacing="1" padding="7" rules="groups" frame="box"/>
  </style>
<colspecs columns="4">
<colgroup>
<colspec name="1" width="15" type="string"/>
<colspec name="2" width="10" align="right" type="string"/>
<colspec name="3" width="16" type="string"/>
<colspec name="4" width="10" align="right" type="string"/>
</colgroup>
</colspecs>
<output-head>
<row>
<header type="string" class="Header" row="1" column="1" column-end="4">
  <style>
    <span columns="4"/>
  </style>
<value>Moments</value>
</header>
</row>
</output-head>
<output-body>

... more tagged output ...

<data raw-value="P8jU/fO2RaI=" name="Low" type="double" class="Data" row="8" column="1">
<value>0.194</value>
</data>
<data raw-value="QEIAAAAAAAA=" name="LowObs" type="double" class="Data" row="8" column="2">
<value>36</value>
</data>
<data raw-value="QDboMSbpeNU=" name="High" type="double" class="Data" row="8" column="3">
<value>22.907</value>
</data>
<data raw-value="QEiAAAAAAAA=" name="HighObs" type="double" class="Data" row="8" column="4">
<value>49</value>
</data>
</row>
</output-body>
</output-object>
</output>
</leaf>
</branch>
</branch>
<footnote class="SystemFooter" toc-level="1">^{super *}This is a  ^S={foreground=black}footnote.</footnote>
</proc>
</body>
</odsxml>
```

Example 2: Creating an XML File and a DTD

ODS features:
- ODS LISTING statement:
  - Action:
    - CLOSE
- ODS MARKUP statement:
  - Actions:
    - CLOSE
  - Options:
    - BODY=
    - FRAME=
    - TAGSET=

Other SAS features:
- PROC UNIVARIATE
- TITLE statement

Data Set:
See “Creating the StatePop Data Set” on page 881

Program Description The following ODS MARKUP example creates an XML file and its Document Type Definition (DTD) related information document from PROC UNIVARIATE output.

Program

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create XML output and a DTD. The ODS MARKUP BODY= statement creates an XML file. The FRAME= option specifies that you want the DTD in a frame file, and the TAGSET= option specifies that you want the default tagset, which is XML.

```
ods markup body='statepop.xml'
           frame='statepop.dtd' tagset=default;
```

Generate the statistical tables for the analysis variables. The UNIVARIATE procedure calculates univariate statistics for numeric variables in the StatePop data set. The VAR statement specifies the analysis variables and their order in the output. The TITLE statement specifies a title for the output object.

```
proc univariate data=statepop;
   var citypop_90 citypop_80;
title 'US Census of Population and Housing';
run;
```

Close the MARKUP destination. The ODS MARKUP CLOSE statement closes the MARKUP destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```
ods markup close;
```

Output This DTD specifies how the markup tags in a group of SGML or XML documents should be interpreted by an application that displays, prints, or otherwise processes the documents.

Output 5.2 DTD Created by the ODS MARKUP Statement

```
<!ELEMENT odsxml (head?,body)>

<!ELEMENT head (meta|css)*>

<!ELEMENT body ((label|page)*|proc)+>

<!ELEMENT meta EMPTY>
<!ATTLIST meta
          operator  CDATA  #IMPLIED
          author    CDATA  #IMPLIED>

<!ELEMENT css EMPTY>
<!ATTLIST css
          file  CDATA  #IMPLIED>

<!ELEMENT label EMPTY>
<!ATTLIST label
          name      ID    #IMPLIED>

<!ELEMENT proc (title|proc-title|note|page|label|style|branch|output)*>
<!ATTLIST proc
          class    CDATA    #IMPLIED>

... more tagged output ...

<!ELEMENT br EMPTY>
<!ELEMENT page EMPTY>

<!ELEMENT b (#PCDATA|it|b|ul)*>
<!ELEMENT ul (#PCDATA|it|b|ul)*>
<!ELEMENT it (#PCDATA|it|b|ul)*>
<!ELEMENT style (span|align|border)*>

<!ELEMENT span EMPTY>
<!ATTLIST span
          columns    CDATA    #IMPLIED
          rows       CDATA    #IMPLIED>

<!ELEMENT align EMPTY>
<!ATTLIST align
          horiz  (left|center|right|justify)    "left">

<!ELEMENT border EMPTY>
<!ATTLIST border
          rules    (none|groups|rows|cols|all)    #IMPLIED
          frame    (void|above|below|hsides|lhs|rhs|vsides|box|border) #IMPLIED
          padding  CDATA                          #IMPLIED
          spacing  CDATA                          #IMPLIED>
```

Example 3: Creating Multiple Markup Output

ODS features:

ODS LISTING statement:

Action:

CLOSE
ODS CSVALL statement:
Options:
BODY=
ODS MARKUP statement:
Options:
BODY=
TAGSET=
TITLE=

Other SAS features:
OPTIONS statement
PROC PRINT
TITLE statement

Data set:
See “Creating the Grain_Production Data Set” on page 878.

Program Description The following ODS example creates two different types of markup output from the same procedure output. To create two markup outputs requires two ODS destinations. Because ODS MARKUP is considered one destination, you cannot specify two tagsets without the use of the ID= option. However, you can specify one output using ODS MARKUP. You can then specify the other output using ODS syntax in which the tagset is the destination.

Program

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources. The OPTIONS statement specifies that only fifteen observations be used.

```
ods listing close;
options obs=15;
```

Create tabular output. The ODS CSVALL statement produces tabular output with titles that contain columns of data values that are separated by commas

```
ods csvall body='procprintcsvall.csv';
```

Create CHTML output. The ODS MARKUP TAGSET=CHTML statement produces compact, minimal HTML output that does not use style information, and a hierarchical table of contents. The TITLE= option specifies the text that will appear in the browser window title bar.

```
ods markup tagset=chtml body='procprintchtml.html'
   (title= 'This Text Identifies Your Content.');
```

Print the data set. The PRINT procedure prints the data set Grain_Production. The TITLE statement specifies the title.

```
title 'Leading Grain-Producing Countries';
proc print data=grain_production;
```

```
run;
```

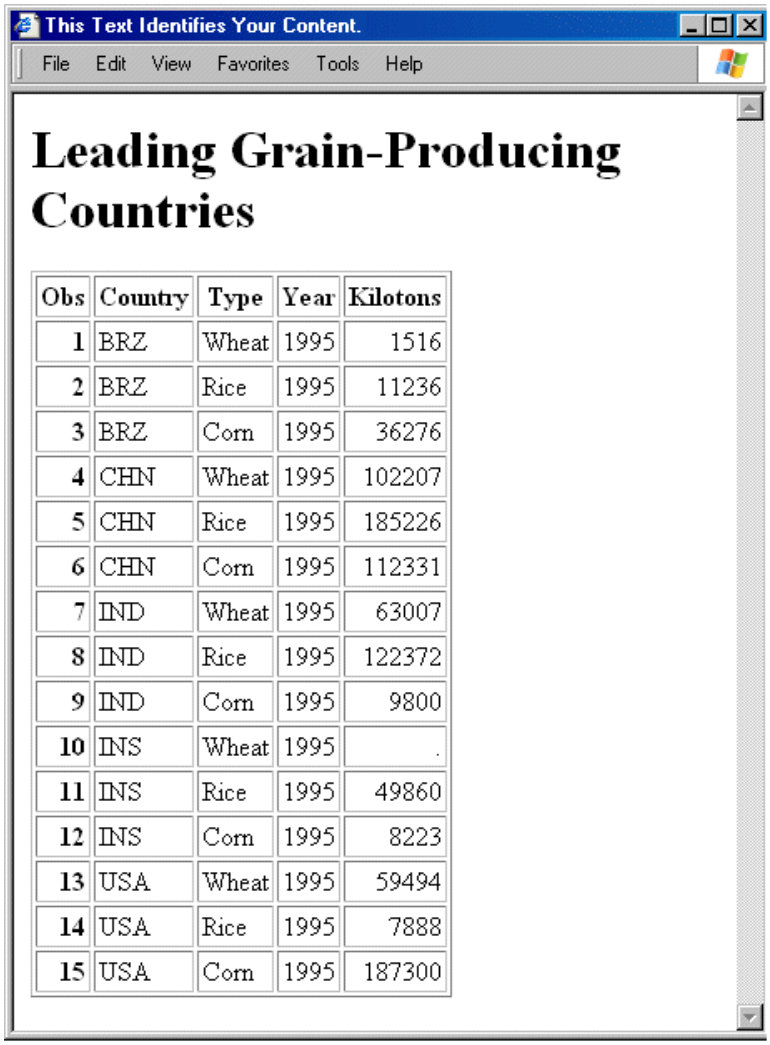
Close the open destinations so that you can view or print the output. The ODS CSVALL CLOSE statement closes the CSVALL destination and all of the files that are associated with it. The ODS MARKUP TAGSET=CHTML L CLOSE statement closes the MARKUP destination and all of the files that are associated with it. You must close the destinations before you can view the output with a browser or before you can send the output to a physical printer.

```
ods csvall close;
ods markup tagset=chtml close;
```

Output

Display 5.7 CHTML Output

The following output was created by specifying the MARKUP TAGSET=CHTML statement. The text "This Text Identifies Your Content." was specified by the TITLE= option.



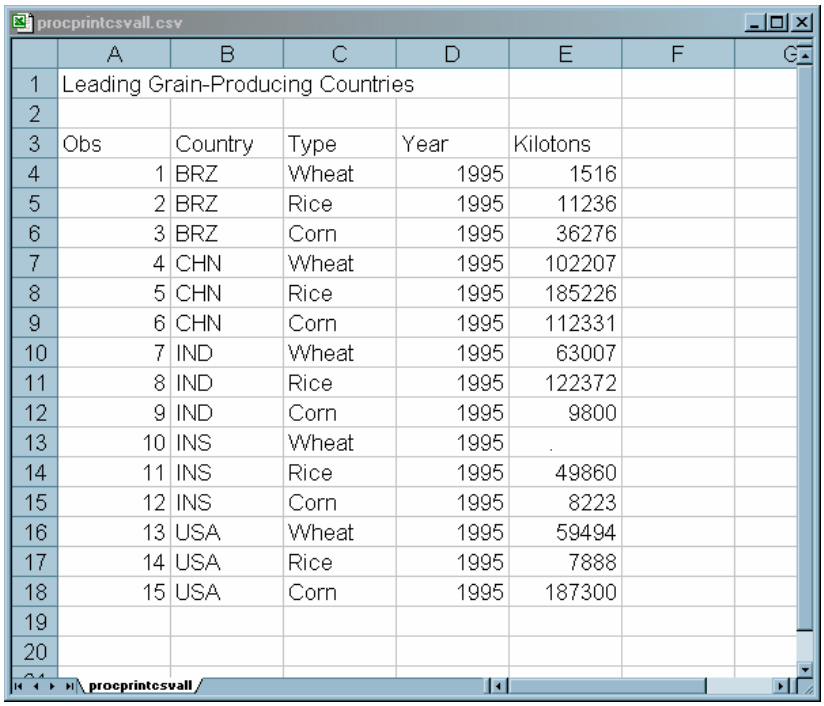
Leading Grain-Producing Countries

Obs	Country	Type	Year	Kilotons
1	BRZ	Wheat	1995	1516
2	BRZ	Rice	1995	11236
3	BRZ	Corn	1995	36276
4	CHN	Wheat	1995	102207
5	CHN	Rice	1995	185226
6	CHN	Corn	1995	112331
7	IND	Wheat	1995	63007
8	IND	Rice	1995	122372
9	IND	Corn	1995	9800
10	INS	Wheat	1995	.
11	INS	Rice	1995	49860
12	INS	Corn	1995	8223
13	USA	Wheat	1995	59494
14	USA	Rice	1995	7888
15	USA	Corn	1995	187300

Display 5.8 CSVALL Output Viewed in Microsoft Excel

The following output was created by specifying the ODS CSVALL statement.

Note: Note that you cannot specify ODS MARKUP TAGSET=CSVALL and ODS MARKUP TAGSET=CHTML together, or ODS CSVALL and ODS CHTML together. △



procprintcsvall.csv

	A	B	C	D	E	F	G
1	Leading Grain-Producing Countries						
2							
3	Obs	Country	Type	Year	Kilotons		
4	1	BRZ	Wheat	1995	1516		
5	2	BRZ	Rice	1995	11236		
6	3	BRZ	Corn	1995	36276		
7	4	CHN	Wheat	1995	102207		
8	5	CHN	Rice	1995	185226		
9	6	CHN	Corn	1995	112331		
10	7	IND	Wheat	1995	63007		
11	8	IND	Rice	1995	122372		
12	9	IND	Corn	1995	9800		
13	10	INS	Wheat	1995	.		
14	11	INS	Rice	1995	49860		
15	12	INS	Corn	1995	8223		
16	13	USA	Wheat	1995	59494		
17	14	USA	Rice	1995	7888		
18	15	USA	Corn	1995	187300		
19							
20							

procprintcsvall

Example 4: Specifying Tagset Names As ODS Destinations When you specify tagsets and two-level tagset names as destinations, you can open and close multiple destinations, producing multiple markup output. For example:

```
ods htmlcss body='test1.html';
ods phtml body='test2.html';
ods chtml body='test3.html';
ods markup body='test1.xml';
ods tagsets.event_map  body='test2.xml';
...more SAS statements...
ods htmlcss close;
...more SAS statements...
ods chtml close;
...more SAS statements...
ods _all_ close;
```

Example 5: Including Multiple Cascading Style Sheets in One HTML Document

ODS features:

ODS LISTING statement:

Action:

CLOSE

ODS HTML statement:

Actions:
CLOSE
Options:
BODY=
STYLESHEET= option
URL= suboption

Other SAS features:
OPTIONS statement
PROC PRINT
TITLE statement

Data set:
See “Creating the Grain_Production Data Set” on page 878 .

Program Description The following example creates one HTML document and two style sheets which are included in the HTML document. The URLs are created in the order specified by the URL= suboption.

Program

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources. The OPTIONS statement specifies that only fifteen observations be used.

```
ods listing close;
options obs=15;
```

Create the HTML output and two style sheets. The ODS HTML statements opens the HTML destination and creates HTML output. The STYLESHEET= option places the style information for the HTML output into two external files. The URL= suboption specifies a URL for the two files, File1.css and File2.css. ODS uses these URLs (instead of the filename) in all the links and references that it creates and that point to those files.

```
ods html body='StylesheetExample.html'
   stylesheet=(url='/css/file1.css /css/file2.css');
```

Print the data set. The PRINT procedure prints the data set Grain_Production. The TITLE statement specifies the title.

```
proc print data=grain_production;
title 'Leading Grain-Producing Countries';
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```
ods html close;
```

Output

Display 5.9 HTML Code

The two links to the style sheets that the STYLESHEET= option creates are at the bottom of the partial output. The links are created in the order that they were specified by the URL= suboption.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta name="Generator" content="SAS Software, see www.sas.com" sasversion="9.2">
<meta http-equiv="Content-type" content="text/html; charset=windows-1252">
<title>SAS Output</title>
<style type="text/css">
<!--
.l {text-align: left }
.c {text-align: center }
.r {text-align: right }
.d {text-align: "." }
.t {vertical-align: top }
.m {vertical-align: middle }
.b {vertical-align: bottom }
TD, TH {vertical-align: top }
-->
</style>
<link rel="stylesheet" type="text/css" href="/css/file1.css">
<link rel="stylesheet" type="text/css" href="/cssfile2.css">
```

Example 6: Applying a CSS File to ODS Output

ODS features:

- ODS HTML statement:
  - BODY= option
  - CSSSTYLE= option:
    - *media-type* suboption
  - TEXT= options
- ODS PDF statement:
  - BODY= option
  - CSSSTYLE= option:
    - *media-type* suboption
  - STARTPAGE= option
  - TEXT= option
- ODS RTF statement:
  - BODY= option
  - CSSSTYLE= option:
    - *media-type* suboption
  - TEXT= options

Other SAS features:

- PROC CONTENTS

Program Description The following program applies a style sheet created in a CSS file to HTML, PDF, and RTF output. Because the CSS file has media blocks with additional information for screen and print media types, you can specify that each output destination use the additional style information for a specific media type.

Program

Example CSS file. The following code is an example of the external CSS file StyleSheet.css. There are two media types specified in this program, Print and Screen.

```
.body {
    background-color: white;
    color: black;
    font-family: times, serif;
}
.header, .rowheader, .footer, .rowfooter, .data {
    border: 1px black solid;
    color: black;
    padding: 5px;
    font-family: times, serif;
}
.header, .rowheader, .footer, .rowfooter {
    background-color: #a0a0a0;
}
.table {
    background-color: #dddddd;
    border-spacing: 0;
    border: 1px black solid;
}
.proctitle {
    font-family: helvetica, sans-serif;
    font-size: x-large;
    font-weight: normal;
}

@media screen {

.header, .rowheader, .footer, .rowfooter,{
    color: white;
    background-color: green;}
.table {
    background-color: yellow;
    border-spacing: 0;
    font-size: small
    border: 1px black solid;

}
}
@media print {

.header, .rowheader, .footer, .rowfooter,{
    color: white;
    background-color: blue;
    padding: 5px;
}
.data {
    font-size: small;
}
}
```

Apply the CSS file to your output. The CSSSTYLE= option on the ODS HTML, ODS RTF, and ODS PDF statements applies the CSS file StyleSheet.css to the output for each destination. Because the *media-type* Screen is specified with the CSSSTYLE= option in the ODS HTML statement, the HTML output has the style information in the Screen media type block applied to it in addition to the style information that is outside of any media blocks. Similarly, the RTF output uses the additional information from the Print media block. The PDF output uses all of the code in the CSS file, because both Print and Screen are specified.

```
options nodate pageno=1 linesize=80 pagesize=40 obs=10;

ods html file="StyleSheet.html" cssstyle='stylesheet.css'(screen) text="Style Sheet Using Scr

ods rtf file="StyleSheet.rtf" cssstyle='stylesheet.css'(print) text="Style Sheet Using Print

ods pdf file="StyleSheet.pdf" cssstyle='stylesheet.css'(print screen) STARTPAGE=no text="Styl
```

View the contents of the SAS data set. The CONTENTS procedure shows the contents of the SAS data set SasHelp.Class.

```
proc contents data=sashelp.class;
run;
```

Close the open destinations. The ODS _ALL_ CLOSE statement closes all open destinations and the files that are associated with them. If you do not close the destinations, then you will not be able to view the files.

```
ods _all_ close;
```

Output

Display 5.10 HTML Output Using Both a Style Sheet with Screen Media Type

The yellow and green background colors, the white font color, the font size and border information all come from the Screen media block. All other style information comes from the code outside of the media blocks. No information from the Print media block is used.

Style Sheet Using Screen Media Type

The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	**Observations**	19
Member Type	DATA	**Variables**	5
Engine	V9	**Indexes**	0
Created	Thursday, May 19, 2005 03:25:02 PM	**Observation Length**	40
Last Modified	Thursday, May 19, 2005 03:25:02 PM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
File Name	C:\SASv9\sasgen\dev\mva-v920\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0201B0
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes			
#	**Variable**	**Type**	**Len**
3	Age	Num	8
4	Height	Num	8

Display 5.11 RTF Output Using a Style Sheet with Print Media Type

The white font, small font size, cell padding, and the blue background color all come from the Print media block. All other style information comes from the code outside of the media blocks. No information from the Screen media block is used.

The CONTENTS Procedure

Style Sheet Using Print Media Type

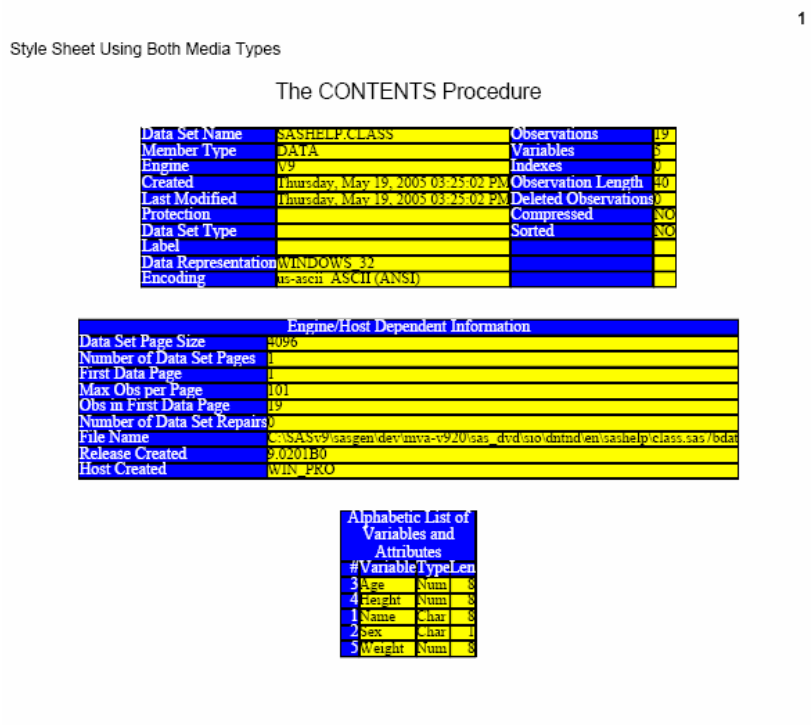
Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Thursday, May 19, 2005 03:25:02 PM	Observation Length	40
Last Modified	Thursday, May 19, 2005 03:25:02 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
File Name	C:\SASv9\sasgen\dev\mva-v920\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0201B0
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Display 5.12 PDF Output Using Both a Style Sheet with both Print and Screen Media Types

The PDF output uses all of the style information in the CSS file, including the information from both media types. However, both the Print and Screen media blocks have a background color specified for row and column headings. The blue background color is picked up because it is specified last.



1

Style Sheet Using Both Media Types

The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Thursday, May 19, 2005 03:25:02 PM	Observation Length	40
Last Modified	Thursday, May 19, 2005 03:25:02 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
File Name	C:\SASv9\sasgen\dev\mva-v920\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0201B0
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Example 7: Using the DOC Suboption to Get ODS TAGSETS.HTMLPANEL Information

ODS features:
    ODS TAGSETS.HTMLPANEL statement:
        Action:
            CLOSE
        Options:
            OPTIONS
                (DOC=”HELP”)
            FILE=

Other SAS features:
    PROC PRINT

Program Description The following example prints to the SAS log the OPTIONS suboptions and a description of each available suboption.

Program

Print information about the OPTIONS suboptions to the SAS log file. Specifying the OPTIONS suboption (DOC='HELP') prints Help for the ODS TAGSETS.HTMLPANEL statement suboptions to the SAS log file. The FILE= option prints the data results to an RTF file named Help.rtf.

```
ods tagsets.panel file='Help.html' options (doc="help");
```

Print the data set SASHELP.CLASS. The PROC PRINT statement prints the SASHELP.CLASS data set.

```
proc print data=Sashelp.Class;
run;
```

Close all destinations. Close the ODS TAGSETS.HTMLPANEL destination and any other open destinations. This statement also closes all the files that are associated with each open destination. If you do not close a destination, then you cannot view the files in a browser window.

```
ods _all_ close;
```

SAS Log Output Specify the "DOC='help' suboption to print all of the OPTIONS suboptions and information about each of the suboptions to the SAS log.

ODS OUTPUT Statement

Produces a SAS data set from an output object and manages the selection and exclusion lists for the OUTPUT destination.

Valid: anywhere

Category: ODS: SAS Formatted

Syntax

ODS OUTPUT *action*;

ODS OUTPUT *data-set-definition(s)*;

Actions

An *action* can be one of the following:

CLEAR
sets the list for the OUTPUT destination to EXCLUDE ALL.

CLOSE
closes the OUTPUT destination. When an ODS destination is closed, ODS does not send output to that destination. Closing a destination frees some system resources.

SHOW
writes to the SAS log the current selection or exclusion list for the OUTPUT destination. If the list is the default list (EXCLUDE ALL), then SHOW also writes the current overall selection or exclusion list.

Required Arguments

data-set-definition
provides instructions for turning an output object into a SAS data set. ODS maintains a list of these definitions. This list is the selection list for the OUTPUT

destination. For information about how ODS manages this list, see "Selection and Exclusion Lists" on page 34. Each *data-set-definition* has the following form:

output-object-specification<=*data-set*>

output-object-specification
has the following form:

output-object<(MATCH_ALL<=*macro-var-name*> PERSIST=PROC | RUN)>

output-object
identifies one or more output objects to turn into a SAS data set.

To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. For more information, see the ODS TRACE statement"ODS TRACE Statement" on page 317. Output Objects can be specified as the following:

- a full path. For example,

```
Univariate.City_Pop_90.TestsForLocation
```

  is the full path of the output object.
- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is

```
Univariate.City_Pop_90.TestsForLocation
```

  then the partial paths are:

```
City_Pop_90.TestsForLocation
TestsForLocation
```

- a label that is enclosed in quotation marks.
  For example,

```
"Tests For Location"
```

- a label path. For example, the label path for the output object is

```
"The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
```

  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement. △
- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is

```
"The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
```

  then the partial label paths are:

```
"CityPop_90"."Tests For Location"
"Tests For Location"
```

- a mixture of labels and paths.
- any of the partial path specifications, followed by a pound sign (#) and a number. For example, **`TestsForLocation#3`** refers to the third output object that is named **`TestsForLocation`**.

Tip: To create multiple data sets from the same output object, list the output object as many times as you want. Each time that you list the output object, specify a different data set.

MATCH_ALL=<*macro-var-name*>
creates a new data set for each output object. For an explanation of how ODS names these data sets, see the discussion of data-set on page 186.

macro-var-name
specifies the macro variable where a list of all the data sets that are created are stored. Thus, if you want to concatenate all the data sets after the PROC step, then you can use the macro variable to specify all the data sets in a DATA step.

Tip: The MATCH_ALL option is not needed to merge conflicting output objects into one data set.

CAUTION:
A data set that is produced by SAS 9.1 without MATCH_ALL will not necessarily be identical to a data set produced by SAS 9.0 with MATCH_ALL and then concatenated in a DATA step. With SAS 9.0, merging dissimilar output objects with the MATCH_ALL option could result in missing columns or truncated variables. With SAS 9.1, these restrictions do not apply. For more information about merging output objects, see "Merging Dissimilar Output Objects into One Data Set" on page 187. △

PERSIST=PROC | RUN
determines when ODS closes any data sets that it is creating, and determines when ODS removes output objects from the selection list for the OUTPUT destination.

PROC
maintains the list of definitions even after the procedure ends, until you explicitly modify it. To modify the list, use ODS OUTPUT with one or more *data-set-specifications*. To set the list for the OUTPUT destination to EXCLUDE ALL, use the following statement:

```
ods output clear;
```

RUN
maintains the list of definitions and keeps open the data sets that it is creating even if the procedure or DATA step ends, or until you explicitly modify the list.

See also: "How ODS Determines the Destinations for an Output Object" on page 35

data-set
names the SAS output data set. You can use a one-level or two-level (with a libref) name.

If you are creating a single data set, then the ODS OUTPUT statement simply uses the name that you specify. If you are creating multiple data sets with MATCH_ALL, then the ODS OUTPUT statement appends numbers to the name. For example, if you specify **`test`** as *data-set* and you create three data sets, then ODS names the first data set **`test`**. The additional data sets are named **`test1`** and **`test2`**.

Note: If you end the filename with a number, then ODS begins incrementing the name of the file with that number. For example, if you specify **`may5`** as *data-set* and you create three data sets, then ODS names the first data set **`may5`**. The additional data sets are named **`may6`** and **`may7`**. △

Default: If you do not specify a data set, then ODS names the output data set DATA*n*, where *n* is the smallest integer that makes the name unique.

Tip: You can specify data set options in parentheses immediately after *data-set*.

NOWARN
suppresses the warning that an output object was requested but not created.

SHOW
functions just like the ODS SHOW statement except that it writes only the selection or exclusion list for the OUTPUT destination.

Details

Merging Dissimilar Output Objects into One Data Set By default, the ODS OUTPUT statement puts all output objects that have the same *output-path* into one SAS data set, regardless of any conflicting variables in the output objects. Variables created by a later output object will get a value of missing in the observations created by the earlier output object. Variables created by an earlier output object that do not exist in a subsequent output object will get a value of missing in the observations added by the later output object. If a variable created by an output object has a different type than a variable with the same name created by an earlier output object, it will be added to the output data set using a new name formed by adding a numeric suffix.

Examples

Example 1: Creating a Combined Output Data Set

ODS features:
- ODS _ALL_ CLOSE statement
- ODS HTML statement:
  - BODY=
  - CONTENTS=
  - FRAME=
  - PAGE=
- ODS LISTING statement:
  - CLOSE
- ODS OUTPUT statement

Other SAS features:
- PROC FORMAT
- PROC PRINT
- PROC TABULATE
- KEEP= data set option

Data Sets:
See “Creating the Energy Data Set” on page 875.

Program Description This example routes two output objects that PROC TABULATE produces to both the OUTPUT destination and the HTML destination. The result is two output objects that are combined by the ODS OUTPUT statement to create an output data set formatted as HTML output by the ODS HTML statement.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program

Format the variables Region, Division, and Type. PROC FORMAT creates formats for **Region**, **Division**, and **Type**.

```
proc format;
   value regfmt 1='Northeast'
                2='South'
                3='Midwest'
                4='West';
   value divfmt 1='New England'
                2='Middle Atlantic'
                3='Mountain'
                4='Pacific';
   value usetype 1='Residential Customers'
                 2='Business Customers';
run;
```

Do not produce listing output. The ODS LISTING statement closes the LISTING destination to conserve resources. Otherwise, output would be written to the LISTING destination by default.

```
ods listing close;
```

Create the SAS output data set. The ODS OUTPUT statement creates the SAS data set EnergyOutput from the output objects that PROC TABULATE produces. The name of each output object is **Table**. You can determine the name of the output objects by using the ODS TRACE ON statement. For information about the ODS TRACE statement, see "ODS TRACE Statement" on page 317.

Specify the variables that you want to be written to the output SAS data set. The KEEP= data set option limits the variables in the output data set EnergyOutput to **Region**, **Division**, **Type**, and **Expenditures_sum**. The variable name **Expenditures_sum** is generated by PROC TABULATE to indicate that the **sum** statistic was generated for the **Expenditures** variable.

```
ods output Table=energyoutput(keep=region division type expenditures_sum);
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC TABULATE is sent to the body file. FRAME=, CONTENTS=, and PAGE= create a frame that includes a table of contents and a table of pages that link to the contents of the body file. The body file also appears in the frame.

```
ods html body='your_body_file.html'
        frame='your_frame_file.html'
     contents='your_contents_file.html'
        page='your_page_file.html';
```

Create output data sets and an HTML report. This PROC TABULATE step creates two output objects named **Table**, one for each BY group, and adds them to the EnergyOutput data set. Because the HTML destination is open, ODS writes the output to the body file.

```
proc tabulate data=energy format=dollar12.;
   by region;
   class division type;
   var expenditures;
   table division,
         type*expenditures;

   format region regfmt. division divfmt. type usetype.;
   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

Close the current body file and open a new file. The ODS HTML BODY= statement closes the original body file and opens a new one. The contents, page, and frame files remain open. The contents and page files will contain links to both body files.

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC TABULATE is sent to the body file. FRAME=, CONTENTS=, and PAGE= create a frame that includes a table of contents and a table of pages that link to the contents of the body file. The body file also appears in the frame.

```
ods html body='your_body_file_2.html';
```

Print the combined data set. This PROC PRINT step prints the data set EnergyOutput that contains both BY groups. The output is added to the current body file, **your_body_file_2.html**.

```
proc print data=energyoutput noobs;
   title 'Combined Output Data Set';
run;
```

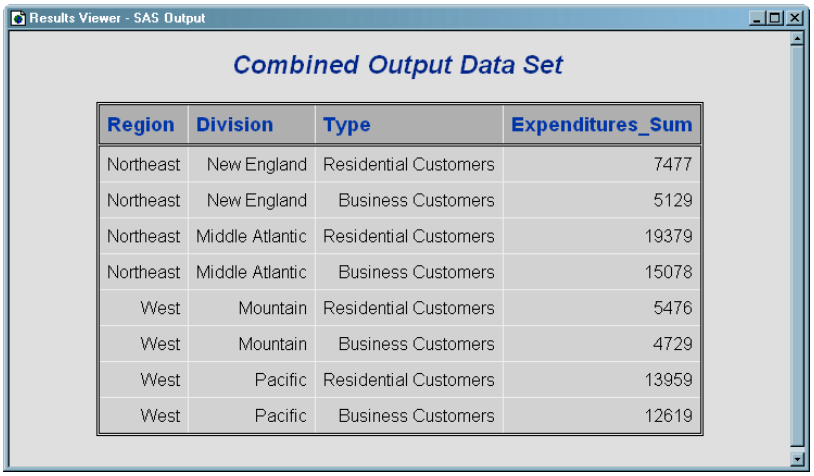
Close all of the open destinations. The ODS _ALL_ CLOSE statement closes all open ODS output destinations. To return ODS to its default setup, the ODS LISTING statement opens the LISTING destination.

```
ods _all_ close;
ods listing;
```

HTML Output

Display 5.13 Combined Data Set

The following HTML output shows the output DATA set that is created by the ODS OUTPUT statement.


Results Viewer - SAS Output

Combined Output Data Set

Region	Division	Type	Expenditures_Sum
Northeast	New England	Residential Customers	7477
Northeast	New England	Business Customers	5129
Northeast	Middle Atlantic	Residential Customers	19379
Northeast	Middle Atlantic	Business Customers	15078
West	Mountain	Residential Customers	5476
West	Mountain	Business Customers	4729
West	Pacific	Residential Customers	13959
West	Pacific	Business Customers	12619

Display 5.14 Output Objects Created by PROC TABULATE

The following output shows the two separate BY groups that are created by the TABULATE procedure.

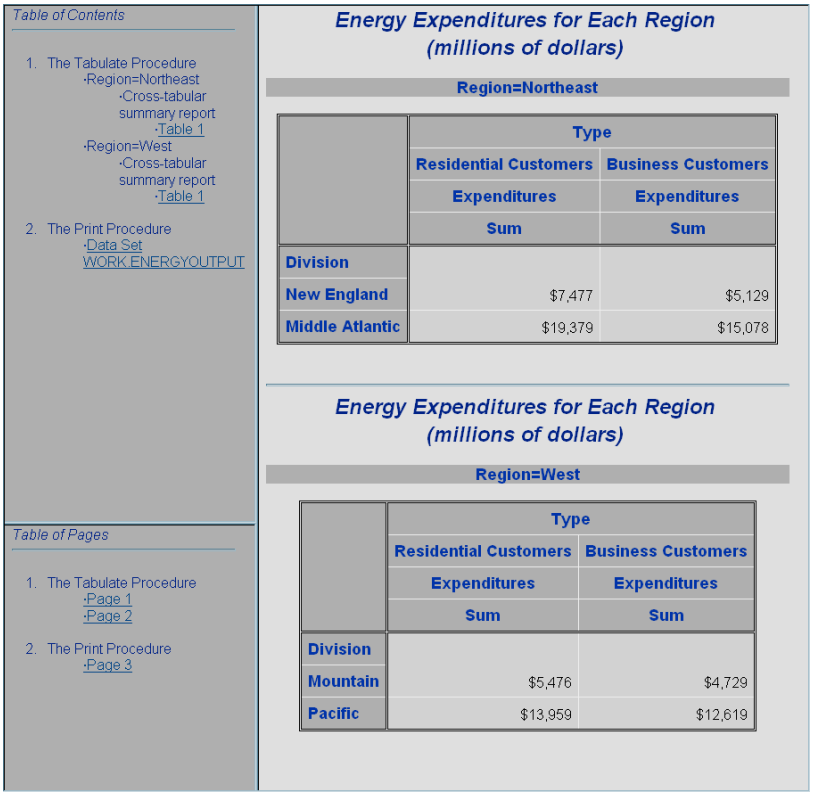


Table of Contents

1. The Tabulate Procedure
   - Region=Northeast
     - Cross-tabular summary report
       - Table 1
   - Region=West
     - Cross-tabular summary report
       - Table 1
2. The Print Procedure
   - Data Set WORK.ENERGYOUTPUT

Table of Pages

1. The Tabulate Procedure
   - Page 1
   - Page 2
2. The Print Procedure
   - Page 3

Energy Expenditures for Each Region (millions of dollars)

Region=Northeast

	Type	
	Residential Customers	Business Customers
	Expenditures	Expenditures
	Sum	Sum
Division		
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078

Energy Expenditures for Each Region (millions of dollars)

Region=West

	Type	
	Residential Customers	Business Customers
	Expenditures	Expenditures
	Sum	Sum
Division		
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619

Example 2: Using Different Procedures to Create a Data Set from Similar Output Objects

ODS features:

- ODS HTML statement:
  - BODY=
  - CONTENTS=
  - FRAME=
- ODS OUTPUT statement
- ODS SELECT statement

Other SAS features:

- PROC GLM
- PROC PRINT
- PROC REG

Data set:

See "Creating the Iron Data Set" on page 879.

Program Description This example creates and prints a data set that is created from the parameter estimates that PROC REG and PROC GLM generate. These procedures are part of SAS/STAT software.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program

Set the SAS system options for the listing output. The NODATE option suppresses the display of the date and time in the listing output. The PAGENO= option specifies the starting page number. The PAGESIZE= option specifies the number of lines on an output page. The LINESIZE= option specifies the output line length.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The FRAME= and CONTENTS= options create a frame that includes a table of contents that links to the contents of the body file. The body file also appears in the frame.

```
ods html body='parameter-estimates-body.htm'
         frame='parameter-estimates-frame.htm'
      contents='parameter-estimates-contents.htm';
```

Specify the output objects to be sent to all open ODS destinations. The ODS SELECT statement specifies that output objects named ParameterEstimates should be sent to all open ODS destinations that do not specifically exclude them. The LISTING destination is open by default, and its default list is SELECT ALL. The ODS HTML statement has opened the HTML destination, and its default list is also SELECT ALL. Thus any object that is named ParameterEstimates will go to both these destinations. The PERSIST option specifies that ParameterEstimates should remain in the overall selection list until the list is explicitly modified.

```
ods select ParameterEstimates(persist);
```

Create the IronParameterEstimates data set. The ODS OUTPUT statement opens the OUTPUT destination and creates the SAS data set IronParameterEstimates. By default, the list for the OUTPUT destination is EXCLUDE ALL. This ODS OUTPUT statement puts ParameterEstimates in the selection list for the destination. The PERSIST=PROC option specifies that ParameterEstimates should remain in the overall selection list until the procedure ends or the list is explicitly modified.

```
ods output ParameterEstimates(persist=proc)=IronParameterEstimates;
```

Create the output objects. PROC REG and PROC GLM each produce an output object named ParameterEstimates. Because the data set definition persists when the procedure ends, ODS creates a output object from each one.

```
proc reg data=iron;
   model loss=fe;
title 'Parameter Estimate from PROC REG';
run;
quit;

proc glm data=iron;
   model loss=fe;
title 'Parameter Estimate from PROC GLM';
run;
quit;
```

Enable all open destinations to receive output objects. The ODS SELECT ALL statement sets the lists for all destinations to their defaults so that ODS sends all output objects to the HTML and LISTING destinations. (Without this statement, none of the output objects from the following PROC PRINT steps would be sent to the open destinations.)

```
ods select all;
```

Print the reports. The PROC PRINT steps print the data set that ODS created from PROC REG and PROC GLM. The output from these steps goes to both the HTML and the LISTING destinations. Links to the HTML output are added to the contents file.

```
proc print data=IronParameterEstimates noobs;
title 'PROC PRINT Report of the Data set from PROC REG';
run;
```

Close the OUTPUT and HTML destinations. The ODS _ALL_ CLOSE statement closes all open destinations except for the LISTING destination, which is open by default.

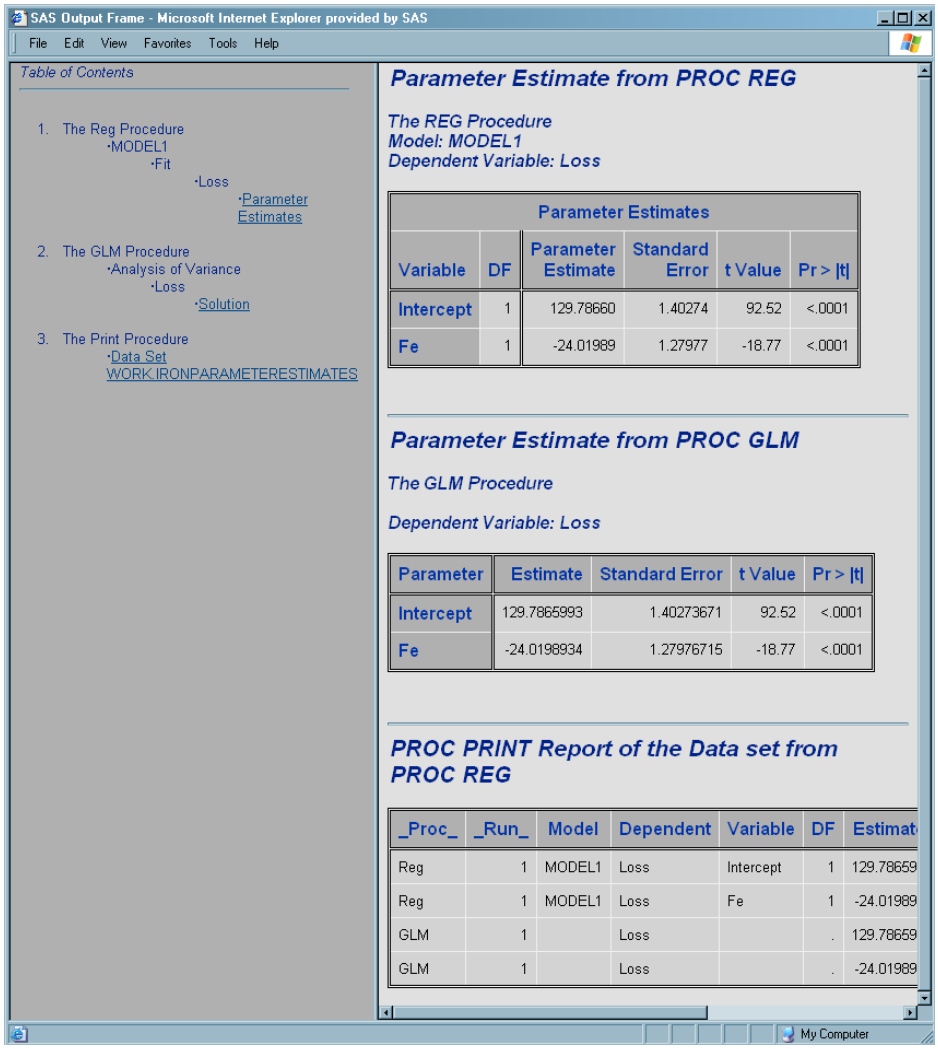
```
ods _all_ close;
```

HTML Output

Display 5.15 HTML Output from the REG, GLM, and PRINT Procedures

The HTML output includes the parameter estimates from PROC REG, the parameter estimates from PROC GLM, and a report of the data set that ODS created from each set of parameter estimates.

The table of contents identifies output objects by their labels. The label for ParameterEstimates in PROC REG is Parameter Estimates. The corresponding label in PROC GLM is Solution. Notice how the column widths in the HTML output are automatically adjusted to fit the data. Compare this layout to the layout of the columns in the listing output.



Listing Output

Output 5.3 Listing Output from the REG, GLM, and PRINT Procedures

```
                    Parameter Estimate from PROC REG                  1

                          The REG Procedure
                            Model: MODEL1
                       Dependent Variable: Loss

                          Parameter Estimates

                       Parameter       Standard
  Variable     DF       Estimate          Error    t Value    Pr > |t|

  Intercept     1      129.78660        1.40274      92.52      <.0001
  Fe            1      -24.01989        1.27977     -18.77      <.0001
```

```
                    Parameter Estimate from PROC GLM                  2

                          The GLM Procedure

Dependent Variable: Loss

                                    Standard
   Parameter         Estimate          Error    t Value    Pr > |t|

   Intercept      129.7865993      1.40273671      92.52      <.0001
   Fe             -24.0198934      1.27976715     -18.77      <.0001
```

```
       PROC PRINT Report of the Data Set Created from PROC GLM and PROC REG 3

  Model  Dependent Variable  DF    Estimate      StdErr  tValue  Probt

  MODEL1   Loss    Intercept  1   129.78660     1.40274   92.52 <.0001
  MODEL1   Loss    Fe         1   -24.01989     1.27977  -18.77 <.0001
```

Example 3: Creating a Data Set with and without The MATCH_ALL Option

ODS features:

- ODS HTML statement:
  - BODY=
- ODS LISTING
- ODS OUTPUT statement:
  - MATCH_ALL
- ODS TRACE statement

Other SAS features:

- PROC PRINT
- PROC REG

Data set:

See “Creating the Model Data Set” on page 879.

Program Description This example illustrates the differences in the data sets created by specifying the MATCH_ALL option and by not specifying the MATCH_ALL option.

The first program creates a merged data set by specifying the MATCH_ALL option. The second program creates a merged data set without specifying the MATCH_ALL option.

The data sets that are printed are parameter estimates that PROC REG generates. The PROC REG procedure is part of SAS/STAT software.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program 1

Do not create listing output. The ODS LISTING statement closes the LISTING destination to conserve resources. Otherwise, output would be written to the LISTING destination by default.

```
ods listing close;
```

Prepare a SAS data set to be created. The ODS OUTPUT statement opens the OUTPUT destination. By default, the list for the OUTPUT destination is EXCLUDE ALL. This ODS OUTPUT statement puts **`SelectionSummary`** in the selection list for the destination.

The MATCH_ALL option produces a SAS data set for each instance of **`SelectionSummary`**. The name of the first data set is Summary, and the name of the second data set is Summary1 . ODS stores a list of these names in the macro variable **`list`**. This variable is used later in the example to combine the data sets.

```
ods output SelectionSummary(match_all=list) = summary;
title1 'Using the MATCH_ALL Option Produces Two Data Sets With Different Columns';
```

Create the output objects and view a record of them in the log. PROC REG creates the output objects.

The ODS TRACE statement writes to the SAS log a record of each output object that is created. The ODS TRACE OFF statement represses the printing of the records.

```
ods trace on;
proc reg data=model;
  model r33=a b r4 r8 c d e r23 r24 r29/ selection=forward
            sle=.5 maxstep=3;
  model r33=a b r4 r8 c d e r23 r24 r29/ selection=backward
            sls=0.05 maxstep=3;
run;
ods trace off;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output.

```
ods html body='combined.html';
```

Print the reports. The PROC PRINT steps print the data sets that ODS created from PROC REG. The output from these steps is sent to both the HTML destination.

```
title2 'The First Data Set Has the VARENTERED Column';
proc print data=summary;
run;

title1;
title2 'The Second Data Set Has the VERREMOVED Column';
proc print data=summary1;
run;
```

Create a data set that contains all of the data sets. The DATA set SummaryM combines all the data sets that were created by the ODS OUTPUT statement. The macro variable **`list`** contains the list of data set names.

```
data summarym;
   set &list;
run;
```

Print the merged report and specify the title. The PROC PRINT step prints the merged data set created from the DATA step. The output from this step is sent to the HTML destination.

The TITLE1 statement cancels the first title, and the TITLE2 statements specify a new title for the output.

```
title1;
title2 'The Merged Data Set Has Both Columns';
proc print data=summarym;
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all of the files that are associated with it.

```
ods html close;
```

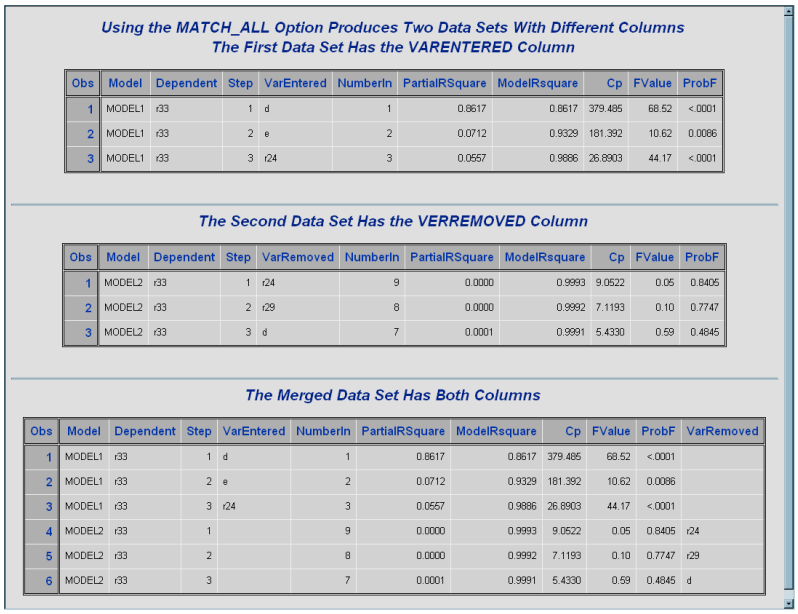
HTML Output

Display 5.16 Three Data Sets Created When Using the MATCH_ALL Option

The First Data Set Created When Using the MATCH_ALL Option This HTML output contains a printed report of the Summary data set created by the ODS OUTPUT statement with the MATCH_ALL option specified. It has no **VERREMOVED** column.

The Second Data Set Created When Using the MATCH_ALL Option This HTML output contains a printed report of the Summary1 data set created by the ODS OUTPUT statement with the MATCH_ALL option specified. It has no **VARENTERED** column.

The Merged Data Set Created When Using the MATCH_ALL Option This HTML output contains a printed report of the SummaryM data set created by the ODS OUTPUT statement with the MATCH_ALL option specified. This is the data set created from Summary and Summary1. It contains both the **VARENTERED** and **VERREMOVED** columns.



Using the MATCH_ALL Option Produces Two Data Sets With Different Columns
The First Data Set Has the VARENTERED Column

Obs	Model	Dependent	Step	VarEntered	NumberIn	PartialRSquare	ModelRsquare	Cp	FValue	ProbF
1	MODEL1	r33	1	d	1	0.8617	0.8617	379.485	68.52	<.0001
2	MODEL1	r33	2	e	2	0.0712	0.9329	181.392	10.62	0.0086
3	MODEL1	r33	3	r24	3	0.0557	0.9886	26.8903	44.17	<.0001

The Second Data Set Has the VERREMOVED Column

Obs	Model	Dependent	Step	VarRemoved	NumberIn	PartialRSquare	ModelRsquare	Cp	FValue	ProbF
1	MODEL2	r33	1	r24	9	0.0000	0.9993	9.0522	0.05	0.8405
2	MODEL2	r33	2	r29	8	0.0000	0.9992	7.1193	0.10	0.7747
3	MODEL2	r33	3	d	7	0.0001	0.9991	5.4330	0.59	0.4845

The Merged Data Set Has Both Columns

Obs	Model	Dependent	Step	VarEntered	NumberIn	PartialRSquare	ModelRsquare	Cp	FValue	ProbF	VarRemoved
1	MODEL1	r33	1	d	1	0.8617	0.8617	379.485	68.52	<.0001	
2	MODEL1	r33	2	e	2	0.0712	0.9329	181.392	10.62	0.0086	
3	MODEL1	r33	3	r24	3	0.0557	0.9886	26.8903	44.17	<.0001	
4	MODEL2	r33	1		9	0.0000	0.9993	9.0522	0.05	0.8405	r24
5	MODEL2	r33	2		8	0.0000	0.9992	7.1193	0.10	0.7747	r29
6	MODEL2	r33	3		7	0.0001	0.9991	5.4330	0.59	0.4845	d

Program 2

Prepare a SAS data set to be created. The ODS OUTPUT statement opens the OUTPUT destination and creates the SAS data set Summary. Because the MATCH_ALL option is not specified, ODS creates one data set that contains all instances of the output object SelectionSummary.

```
ods output SelectionSummary=summary;
title1 'Without the MATCH_ALL Option, ODS Produces a Single Data Set With All
        Of the Columns';
```

Create the output objects and view a record of them in the log. PROC REG creates the output objects.

The ODS TRACE statement writes to the SAS log a record of each output object that is created. The ODS TRACE OFF statement represses the printing of the records.

```
ods trace on;
proc reg data=model;
  model r33=a b r4 r8 c d e r23 r24 r29/ selection=forward
```

```
            sle=.5 maxstep=3;
  model r33=a b r4 r8 c d e r23 r24 r29/ selection=backward
            sls=0.05 maxstep=3;
run;
ods trace off;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output.

```
ods html body='combined2.html';
```

Print the combined data set. The PROC PRINT step prints the merged data set created by ODS. The output from this step is sent to the HTML destination.

```
proc print data=summary;
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all of the files that are associated with it.

```
ods html close;
```

HTML Output

Display 5.17 Using the ODS OUTPUT Statement Without the MATCH_ALL Option to Combine Data Sets

This HTML output contains a printed report of the Summary data set created by the ODS OUTPUT statement without the MATCH_ALL option specified. Note that to merge data sets, you do not have to specify the MATCH_ALL option.

Without the MATCH_ALL Option, ODS Produces a Single Data Set With All Of the Columns

Obs	Model	Dependent	Step	VarEntered	NumberIn	PartialRSquare	ModelRsquare	Cp	FValue	ProbF	VarRemoved
1	MODEL1	r33	1	d	1	0.8617	0.8617	379.485	68.52	<.0001	
2	MODEL1	r33	2	e	2	0.0712	0.9329	181.392	10.62	0.0086	
3	MODEL1	r33	3	r24	3	0.0557	0.9886	26.8903	44.17	<.0001	
4	MODEL2	r33	1		9	0.0000	0.9993	9.0522	0.05	0.8405	r24
5	MODEL2	r33	2		8	0.0000	0.9992	7.1193	0.10	0.7747	r29
6	MODEL2	r33	3		7	0.0001	0.9991	5.4330	0.59	0.4845	d

ODS PACKAGE Statement

The ODS PACKAGE statement opens, adds to, publishes, or closes one SAS Output Delivery System (ODS) package object.

Valid: anywhere

Category: Data Access

Requirement: The destination must specify the PACKAGE option to connect with the package.

Syntax

ODS PACKAGE (<name>) **OPEN** <*options*>;

ODS PACKAGE (<name>) **ADD** FILE=*"file-specification"* |
DATA=*member-specification* MIMETYPE=*"string"*
<PATH=*"path-specification"*><*options*>;

ODS PACKAGE (<name>) **PUBLISH** *transport*
PROPERTIES*(transport-property-1="value-1" … transport-property-n="value-n")*;

ODS PACKAGE (<name>) **CLOSE** <CLEAR>;

Required Arguments

ADD
adds a file or data set to an ODS package using the specified Multipurpose Internet Mail Extensions (MIME) type.

Requirement: When using the ADD argument, you must also use the MIMETYPE=, FILE=, or DATA= arguments to specify a file or data set and a MIME type.

FILE="*file-specification*" <TEXT | BINARY>
specifies the file that you want to add to an ODS package.

file-specification
specifies one of the following:

external-file is the name of an external file to add.

Requirement: You must enclose *external-file* in quotation marks.

fileref is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

TEXT
specifies that the file is a text file.

BINARY
specifies that the file is a binary file.

Default: If you do not specify the TEXT or BINARY values, then file is text if the mimetype is text, and binary if the mimetype is anything else.

Example: Use the following statement to add the Test.Sas file as plain text to the ODS package directory SAS:

```
ods package add file="test.sas" mimetype="text/plain" path="sas/";
```

Restriction: You can use the FILE= argument only with the ADD argument.

Restriction: You cannot add a file and a data set to an ODS package.

DATA=*member-specification*
specifies the data set that you want to add to an ODS package. *member-specification* can be in the form *libname.membername* or *membername*.

Restriction: You can use the DATA= argument only with the ADD argument.

Restriction: You cannot add a file and a data set to an ODS package.

MIMETYPE="*string*"
specifies the Multipurpose Internet Mail Extensions (MIME) type for the file or data set that you are adding to an ODS package.

Restriction: You can use the MIMETYPE= argument only with the ADD argument.

OPEN EXPIRATION= <'*expiration-date*'>
creates the ODS package object to which the ODS destinations can connect. The ODS package object holds the package metadata and tracks the locations of any files that are added to the package metadata.

Example: The following ODS PACKAGE statement opens an unnamed package with an abstract and a description.

```
ods package open abstract="this is my abstract" description="this is
     description";
```

PUBLISH EXPIRATION=<'*expiration-date*'>
builds the ODS package and sends it to the chosen delivery transport.

expiration-date
specifies an expiration date for the package. The date must be a SAS date value.

Requirement: *expiration-date* must be enclosed in quotation marks.

CLOSE
deletes the package object. As long as you have not closed a package, you can publish it as many ways and times as you want.

Tip: Use the CLEAR option to remove files that have been added to the package.

transport
specifies the deliver transport to use with the PUBLISH action. *transport* can be one of the following:

ARCHIVE PROPERTIES(*transport-property-1*="*value-1*"… *transport-property-n*="*value-n*"")
publishes a package to an archive. For a list of transport properties and their values, see the section on transport properties in SAS Integration Technologies Developer's Guide at **http://support.sas.com/rnd/itech/doc9/dev_guide/app/pkgintf/pkg_publ.html**.

Example: The following statement publishes an ODS package to the archive Test.spk:

```
ods package publish archive  properties(archive_path="./"
     archive_name="test.spk");
```

EMAIL PROPERTIES(*transport-property-1*="*value-1*" . . . *transport-property-n*="*value-n*") ADDRESSES("*e-mail-address-1*" . . . "*e-mail-address-n*")
publishes a package to one or more e-mail addresses. For a list of transport properties and their values, see the section on transport properties in *SAS Integration Technologies Developer's Guide* at **http://support.sas.com/rnd/itech/doc9/dev_guide/app/pkgintf/pkg_publ.html**.

Example: The following statement publishes an ODS package to the e-mail addresses your.email@company.com and your.second.email@company.com:

```
ods package publish email  addresses("your.email@company.com"
     "your.second.email@company.com")
    properties(archive_name="testPackage" archive_path="./");
```

QUEUE PROPERTIES(*transport-property-1*="*value-1*" . . . *transport-property-n*="*value-n*") QUEUES("*queue-1*" . . . "*queue-n*")
publishes a package to one or more message queues. For a list of transport properties and their values, see the section on transport properties in *SAS Integration Technologies Developer's Guide* at **http://support.sas.com/rnd/itech/doc9/dev_guide/app/pkgintf/pkg_publ.html**.

SUBSCRIBERS PROPERTIES(*transport-property-1*="*value-1*" . . . *transport-property-n*="*value-n*")
publishes a package to subscribers who are associated with the specified channel. For a list of transport properties and their values, see the section on transport properties in *SAS Integration Technologies Developer's Guide* at **http://support.sas.com/rnd/itech/doc9/dev_guide/app/pkgintf/pkg_publ.html**.

WEBDAV PROPERTIES(*transport-property-1*="*value-1*" . . . *transport-property-n*="*value-n*")
publishes a package to a WebDAV-compliant server. For a list of transport properties and their values, see the section on transport properties in *SAS Integration Technologies Developer's Guide* at **http://support.sas.com/rnd/itech/doc9/dev_guide/app/pkgintf/pkg_publ.html**.

Options

ABSTRACT=*string*
specifies a string for the abstract metadata of the package or file.

Restriction: You can use the ABSTRACT= option only with the ADD or OPEN arguments.

CLEAR
specifies that all files that were automatically added to the package will be removed from the location to which ODS wrote them.

Restriction: You can use the CLEAR option only with the CLOSE argument.

DESCRIPTION=*string*
specifies a string for the description metadata for the package or file.

Restriction: You can use the DESCRIPTION= option only with the ADD or OPEN arguments.

(*name*)
specifies the name of a package. Naming a package enables you to open more than one package at a time. Each destination can connect with any package by specifying the package name in the same way.

Requirement: You must place *name* directly after the PACKAGE keyword in the ODS PACKAGE statement.

Requirement: *name* must be enclosed in parenthesis.

NAMEVALUE="<*name-1*="*value-1*" . . . *name-n*="*value-n*">"
specifies a string of name/value pairs for the name/value metadata on the package or file.

Restriction: The NAMEVALUE= option can be used only with the ADD or OPEN arguments.

PATH="*path-specification*"
places the file or data set at the specified pathname within an ODS package.

Restriction: You can use the PATH= option only with the ADD argument.

Example: Use the following statement to add the Test.Sas file as plain text to the ODS package directory SAS:

```
ods package add file="test.sas" mimetype="text/plain" path="sas/";
```

TEMPLATE=
specifies the name of a package template to use.

Restriction: You can use the TEMPLATE= option only with the ADD or OPEN arguments.

Details

A package is a container for digital content that is generated or collected for delivery to a consumer. ODS packages allow ODS destinations to use the SAS Publishing Framework. An ODS package is an object that contains output files and data sets that are associated with any open ODS destinations. ODS packages hold the package metadata and track the output from any active destinations that connect to it. After the destinations are closed, the package can be published to any of the publish destinations. You can continue to use the package, or you can close it. A package remains active until explicitly closed.

Examples

Example 1: Creating an ODS Package

Program Description: The following example creates a simple ODS package. The package is created in your default directory, if you do not specify a different directory.

Program

Close the LISTING destination and specify graphical options with the GOPTIONS statement.

```
ods listing close;
goptions dev=gif xpixels=480 ypixels=320;
```

Open an ODS package and specify that HTML output be added to the package. The ODS PACKAGE statement opens an ODS package with no name. The PACKAGE option specified by the ODS HTML statement specifies that output from the HTML destination be added to the package.

```
ods package open;
ods html package;
```

Create graphical output with the GPLOT statement and close the HTML destination.

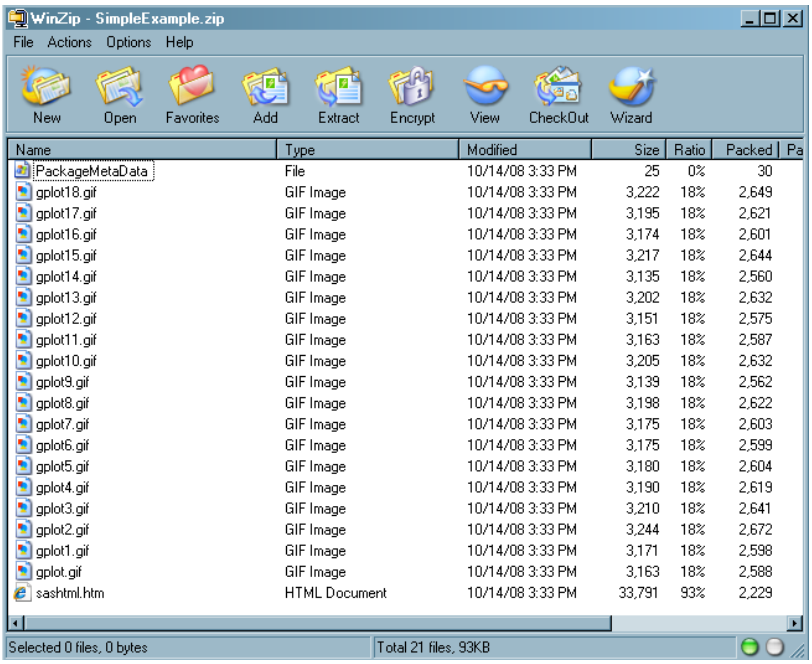
```
 proc gplot data=sashelp.class;
     plot height*weight;
     by name;
 run;
 quit;
ods html close;
```

Build the package and publish it to an archive. The PUBLISH option builds the ODS package. The ARCHIVE property publishes the package to the archive named SimpleExample.zip in the default directory.

```
ods package publish archive properties(archive_name="SimpleExample.zip"
   archive_path="./");
ods package close;
```

Program Output

Display 5.18 Simple ODS Package



Example 2: Listing Package Contents with the ODS DOCUMENT Statement

Program Description: In the following program, PROC DOCUMENT imports the archive SimpleExample.zip into a PROC DOCUMENT package named myPackage.You can then use PROC DOCUMENT to list the contents and details of the package.

Program

Open the LISTING destination.

```
ods listing;
```

Create an ODS document and import SimpleExample.zip. The DOCUMENT procedure creates the ODS document Archive. The IMPORT TO statement imports SimpleExample.zip into the package myPackage. The LIST statement lists all of the levels of Archive.

```
proc document name=archive;
   import archive="SimpleExample.zip" to myPackage;

   list/levels=all;
   run;
```

List the details of the file SasHtml.htm. The DIR statement changes the directory to myPackage. The LIST statement lists the details of SasHtml.htm.

```
   dir myPackage;
   list 'sashtml.htm'n/details;
   run;
quit;
```

Program Output

Output 5.4 Listing of Work.Archive and Details of HTM File

```
                                    The SAS System

Listing of: \Work.Archive\
Order by: Insertion
Number of levels: All

  Obs    Path                                                                  Type
-------------------------------------------------------------------
       1 \myPackage#1                                                          Dir
       2 \myPackage#1\'sashtml.htm'n#1                                         File
       3 \myPackage#1\'gplot.gif'n#1                                           File
       4 \myPackage#1\'gplot1.gif'n#1                                          File
       5 \myPackage#1\'gplot2.gif'n#1                                          File
       6 \myPackage#1\'gplot3.gif'n#1                                          File
       7 \myPackage#1\'gplot4.gif'n#1                                          File
       8 \myPackage#1\'gplot5.gif'n#1                                          File
       9 \myPackage#1\'gplot6.gif'n#1                                          File
      10 \myPackage#1\'gplot7.gif'n#1                                          File
      11 \myPackage#1\'gplot8.gif'n#1                                          File
      12 \myPackage#1\'gplot9.gif'n#1                                          File
      13 \myPackage#1\'gplot10.gif'n#1                                         File
      14 \myPackage#1\'gplot11.gif'n#1                                         File
      15 \myPackage#1\'gplot12.gif'n#1                                         File
      16 \myPackage#1\'gplot13.gif'n#1                                         File
      17 \myPackage#1\'gplot14.gif'n#1                                         File
      18 \myPackage#1\'gplot15.gif'n#1                                         File
      19 \myPackage#1\'gplot16.gif'n#1                                         File
      20 \myPackage#1\'gplot17.gif'n#1                                         File
      21 \myPackage#1\'gplot18.gif'n#1                                         File
                                              The SAS System

Listing of: \Work.Archive\myPackage#1\'sashtml.htm'n#1
Order by: Insertion
Number of levels: 1

              Size
Type        in Bytes         Created             Modified      Symbolic Link    Template
-------------------------------------------------------------------
File              4762  17OCT2008:14:32:52  17OCT2008:14:32:52

Label
-------------------------------------------------------------------
MARKUP, Proc, Gplot, GPLOT, Plot of Height by Weight, GPLOT1, Plot of Height by Weight, GPLOT2,
Plot of Height by Weight, GPLOT3, Plot of Height by Weight, GPLOT4, Plot of Height by Weight,
GPLOT5, Plot of Height by Weight, GPLOT6, Plot of Height by Weigh
```

See Also

Publishing Framework feature of SAS Integrated Technologies
http://support.sas.com/rnd/itech/doc9/dev_guide/publish/index.html

Packages with Publishing Framework
http://support.sas.com/rnd/itech/doc9/dev_guide/publish/package.html

Transport properties in the *SAS Integration Technologies Developer's Guide*
http://support.sas.com/rnd/itech/doc9/dev_guide/app/pkgintf/pkg_publ.html

ODS PATH Statement

Specifies locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them.

Valid: anywhere

Category: ODS: Output Control

Tip: This statement overrides the ODS PATH statement for the duration of a PROC TEMPLATE step.

Tip: You can use the SYSODSPATH automatic macro variable to store the current ODS path. For information on the SYSODSPATH macro variable, see *SAS Macro Language: Reference*.

Syntax

PATH <(APPEND) | (PREPEND) | (REMOVE) > *location(s)*;

PATH *path-argument*;

Required Arguments

location(s)
specifies one or more locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them. ODS searches the locations in the order that they appear on the statement. It uses the first definition that it finds that has the appropriate access mode (read, write, or update) set.

Each *location* has the following form:

<*libref.*>*item-store* <(READ | UPDATE | WRITE)>

<*libref.*>*item-store*
identifies an item store to read from, to write to, or to update. If an item store does not already exist, then the ODS PATH statement will create it.

(READ | UPDATE | WRITE)
specifies the access mode for the definition. The access mode is one of the following:

READ
provides read-only access.

WRITE
provides Write access (always creating a new template store) as well as Read access.

UPDATE
provides Update access (creating a new template store only if the specified one does not exist) as well as Read access.

Default: READ

Default: The general default path is:

SASUSER.TEMPLAT (UPDATE)

SASHELP.TMPLMST (READ)

Note: SAS stores all the definitions that it provides in SASHELP.TMPLMST. △

If you have the RSASUSER SAS system option specified, the default path is:
WORK.TEMPLAT(UPDATE)
SASUSER.TEMPLAT (READ)
SASHELP.TMPLMST (READ)

Note: See the RSASUSER SAS system option in *SAS Language Reference: Dictionary* for more information. △

Interaction: You can use the PATH statement in a PROC TEMPLATE step to temporarily override the ODS PATH statement (see "PATH Statement" on page 416 in PROC TEMPLATE).

Tip: If you want to be able to ignore all definitions that you create, then keep them in their own item stores so that you can leave them out of the list of item stores that ODS searches.

path-argument
specifies the setting or displaying of the ODS path.
path-argument can be one of the following:

RESET
sets the ODS path to the default settings SASUSER.TEMPLAT (UPDATE) and SASHELP.TMPLMST (READ).

SHOW
displays the current ODS path.

VERIFY
sets the ODS path to include only templates supplied by SAS. VERIFY is the same as specifying ODS PATH SASHELP.TMPLMST (READ).

Options

(APPEND | PREPEND | REMOVE)
adds or removes one or more locations to a path.

APPEND
adds one or more locations to the end of a path. When you append a location to a path, all duplicate instances (same name and same permissions) of that item store are removed from the path. Only the last item store with the same name and permissions are kept.

PREPEND
adds one or more locations to the beginning of a path. When you prepend a location with update permissions to a path, all duplicate instances (same name and same permissions) of that item store are removed from the path. Only the first item store with the same name and permissions are kept.

REMOVE
removes one or more locations from a path.

Default: If you do not specify an APPEND, PREPEND, or REMOVE option, then the ODS PATH statement overwrites the complete path.

ODS PCL Statement

Opens, manages, or closes the PCL destination, which produces printable output for PCL (HP LaserJet) files.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS PCL <(<ID=>*identifier*)> <*action*>;

ODS PCL <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS PCL statement without an action or options, then it opens the PCL destination and creates PCL output.

Actions

The following table lists the actions available for the ODS PCL statement. For complete descriptions of actions, see“Actions” on page 219 in the ODS PRINTER statement.

Table 5.20 ODS PCL Action Summary Table

Task	Action
Close the PCL destination and the file that is associated with it	CLOSE
Exclude output objects from the PCL destination	EXCLUDE
Select output objects for the PCL destination	SELECT
Write to the SAS log the current selection or exclusion list for the PCL destination	SHOW

Options

The following table lists the options that are available for the ODS PCL statement. For more detailed descriptions of these options, see “Options” on page 219 in the ODS PRINTER statement.

Table 5.21 ODS PCL Option Summary Table

Task	Option
Apply a specified color scheme to your output	COLOR=
Specify the number of columns to create on each page of output	COLUMNS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify the image resolution for output files	DPI=
Specify the file to write to	FILE=
Open multiple instances of the same destination at the same time	ID=
Create a new file at the specified *starting-point*	NEWFILE=
Specify that the output from the destination be added to an ODS package	PACKAGE
Control page breaks	STARTPAGE=
Specify the style definition to use in writing the PCL output	STYLE=
Insert text into your output	TEXT=
For tables with multiple pages, ensure uniformity from page to page within a single table	UNIFORM

Details

Opening and Closing the PCL Destination You can modify an open PCL destination with many ODS PCL options. However, the FILE= and SAS options will automatically close the open destination that is referred to in the ODS PCL statement, and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

The ODS Printer Family of Statements The ODS PCL statement is part of the ODS printer family of statements. Statements in the printer family open the PCL, PDF, PRINTER, or PS destination, producing output that is suitable for a high-resolution printer. The ODS PDF, ODS PRINTER, and ODS PS statements are also members of the ODS printer family of statements.

See Also

Statements:

"ODS PDF Statement" on page 210
"ODS PRINTER Statement" on page 218
"ODS PS Statement" on page 239
"The Third-Party Formatted Destinations" on page 26

ODS PDF Statement

Opens, manages, or closes the PDF destination, which produces PDF output, a form of output that is read by Adobe Acrobat and other applications.

Valid: anywhere

Category: ODS: Third-Party Formatted

Tip: The PDF driver that SAS uses does not recognize all Microsoft Windows fonts. You must enter any such fonts into the SAS registry in order for SAS to find them. See the SAS registry information in *SAS Language Reference: Concepts*.

Syntax

ODS PDF <(<ID=>*identifier*)> <*action*>;

ODS PDF <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS PDF statement without an action or options, then it opens the PDF destination and creates PDF output.

Actions

The following table lists the actions available for ODS PDF statement. For complete descriptions see “Actions” on page 219 in the ODS PRINTER statement.

Table 5.22 ODS PDF Action Summary Table

Task	Action
Close the PCL destination and the file that is associated with it	CLOSE
Exclude output objects from the PCL destination	EXCLUDE
Select output objects for the PCL destination	SELECT
Write to the SAS log the current selection or exclusion list for the PCL destination	SHOW

Options

The following table lists the options that are available for the ODS PDF statement. For more detailed descriptions of these options, see “Options” on page 219 in the ODS PRINTER statement.

Table 5.23 ODS PDF Option Summary Table

Task	Option
Specify the root name for the anchor tag that identifies each output object in the current file	ANCHOR=
Insert the text string that you specify as the author in the metadata of a file	AUTHOR=
Specify a string to use as the first part of all references that ODS creates in the file	BASE=
Specify whether to generate and display the list of bookmarks for a PDF file	BOOKMARKLIST=
Control the generation of bookmarks in a PDF file	BOOKMARKGEN=
Apply a specified color scheme to your output	COLOR=
Specify the number of columns to create on each page of output	COLUMNS=
Specify the compression of a PDF file. Compression reduces the size of the file	COMPRESS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify the image resolution for output files	DPI=
Specify the file to write to	FILE=
Open multiple instances of the same destination at the same time	ID=
Insert a string of keywords into the output file's metadata	KEYWORDS=
Create a new file at the specified *starting-point*	NEWFILE=
Specify that the output from the destination be added to an ODS package	PACKAGE
Control whether notes are added to a PDF file for items that are associated with the FLYOVER= style attribute	PDFNOTE
Control the level of the expansion of the table of contents in PDF documents	PDFTOC=
Control page breaks	STARTPAGE=
Specify the style definition to use in writing the PDF output	STYLE=
Insert the text string that you specify as the subject in the metadata of a file	SUBJECT=
Insert text into your output	TEXT=

Task	Option
Insert the text string that you specify as the title in the metadata of a file	TITLE=
For multi-page tables, provide uniformity from page to page within a single table	UNIFORM

Details

The ODS Printer Family of Statements The ODS PDF statement is part of the ODS printer family of statements. Statements in the printer family open the PCL, PDF, PRINTER, or PS destination, producing output that is suitable for a high-resolution printer. The ODS PCL, ODS PRINTER, and ODS PS statements are also members of the ODS printer family of statements.

Opening and Closing the PDF Destination You can modify an open PDF destination with many ODS PDF options. However, the FILE= and SAS options will automatically close the open destination that is referred to in the ODS PDF statement, and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

Examples

Example 1: Opening Multiple Instances of the Same Destination at the Same Time

ODS features:
- ODS PDF statement:
  - Options:
    - ID=
    - STYLE=
    - FILE=

Other SAS features:
- PROC FORMAT
- PROC SORT
- PROC REPORT
- NOBYLINE|BYLINE system option
- #BYVAL parameter in titles

Data set:
See “Creating the Grain_Production Data Set” on page 878.

Format:
See “Creating the $CNTRY Format” on page 869.

This example opens multiple instances of the PDF destination to create PDF output. One instance uses the default style definition and the second instance uses the STYLE= option to specify the D3D style definition.

Program

Sort the data set Grain_Production. PROC SORT sorts the data first by values of Year, then by values of Country, and finally by values of Type.

```
proc sort data=grain_production;
   by year country type;
```

```
run;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources. If the destination were left open, then ODS would produce both Listing and PDF output.

```
ods listing close;
```

Create two different PDF output files at the same time. The ODS PDF statement opens the PDF destination and creates PDF output.

The file Grain-1.pdf is created by the first ODS PDF statement. Because no style definition is specified, the default style, Styles.Printer, is used. The PDFTOC=2 option specifies that the table of contents is expanded two levels.

The file Grain-2.pdf is created by the second ODS PDF statement with the ID= option specified. The STYLE= option specifies that ODS use the style definition Brick. The ID= option gives this instance of the PDF destination the name BrickStyle. The PDFTOC=3 option specifies that the table of contents is expanded three levels.

Note: If you do not specify the ID= option, this ODS PDF statement will close the instance of the PDF destination that was opened by the previous ODS PDF statement and open a new instance of the PDF destination. The file Grain-1.pdf will contain no output. △

```
ods pdf file='grain-1.pdf' pdftoc=2;
ods pdf (id=brickstyle) style=brick file='grain-2.pdf' pdftoc=3;
```

Suppress the default BY line, suppress the printing of the date, and use the BY value in a title. The NOBYLINE option suppresses the BY line. The #BYVAL specification inserts the current value of the BY variable Year into the title.

```
options nobyline nodate;
title 'Leading Grain-Producing Countries';
title2 'for #byval(year)';
```

Produce a report. This PROC REPORT step produces a report on grain production. Each BY group produces a page of output.

```
proc report data=grain_production nowindows;
   by year;
   column country type kilotons;
   define country  / group width=14 format=$cntry.;
   define type     / group 'Type of Grain';
   define kilotons / format=comma12.;
   footnote 'Measurements are in metric tons.';
run;
```

Restore the BY line and clear the second title statement. The BYLINE option restores the BY line. The TITLE2 statement clears the second TITLE statement.

```
options byline;
title2;
```

Produce a report that contains one table for each year. The TABLE statement in this PROC TABULATE step has Year as the page dimension. Therefore, PROC TABULATE explicitly produces one table for 1995 and one for 1996.

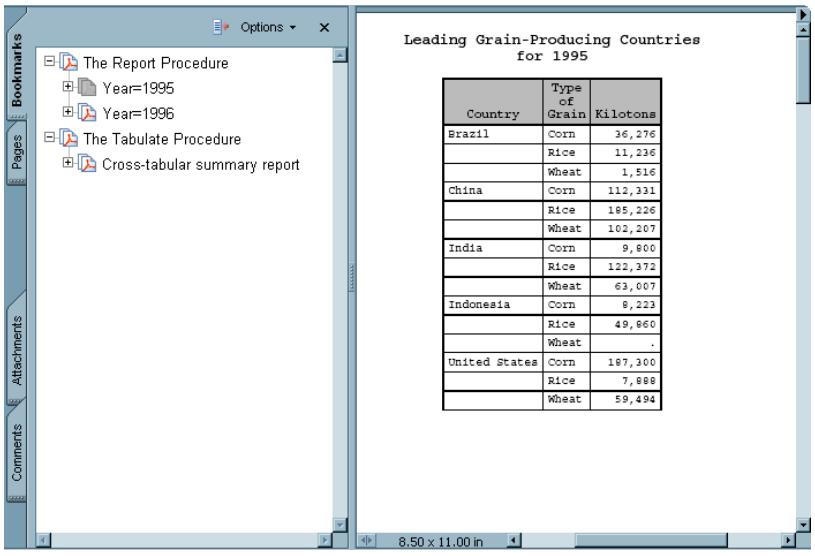
```
proc tabulate data=grain_production format=comma12.;
   class year country type;
   var kilotons;
   table year,
         country*type,
         kilotons*sum=' ' / box=_page_ misstext='No data';
   format country $cntry.;
   footnote 'Measurements are in metric tons.';
run;
```

Close the open destinations so that you can view or print the output. The ODS PDF CLOSE statement closes the first instance of the PDF destination and all of the files that are associated with it. The ODS PDF (ID=d3dstyle) statement closes the second instance of the PDF destination and all of the files that are associated with it. You must close the destinations before you can view the output with a browser or before you can send the output to a physical printer.

```
ods pdf close;
ods pdf(id=d3dstyle) close;
```

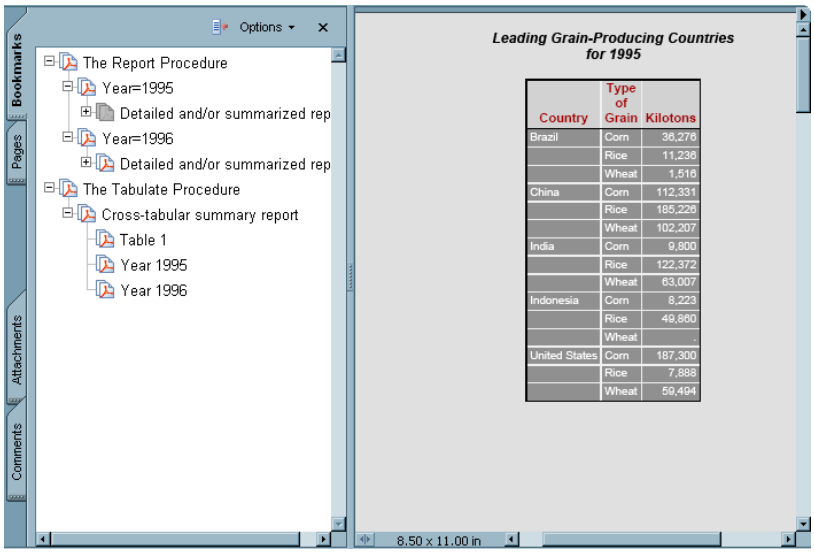
PDF Output

Display 5.19 PDF Output With the Default Style Definition



Country	Type of Grain	Kilotons
Brazil	Corn	36,276
	Rice	11,236
	Wheat	1,516
China	Corn	112,331
	Rice	185,226
	Wheat	102,207
India	Corn	9,800
	Rice	122,372
	Wheat	63,007
Indonesia	Corn	8,223
	Rice	49,860
	Wheat	.
United States	Corn	187,300
	Rice	7,888
	Wheat	59,494

Display 5.20 PDF Output Using Brick Style Definition



See Also

Statements:

"ODS PCL Statement" on page 208
"ODS PRINTER Statement" on page 218
"ODS PS Statement" on page 239
"The Third-Party Formatted Destinations" on page 26

ODS PHTML Statement

Opens, manages, or closes the PHTML destination, which produces simple HTML output that uses twelve style elements and no class attributes for the presentation. Class attributes are used only for the justification.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS PHTML *action*;

ODS PHTML <*option(s)*>;

Without an Action or Options

If you use the ODS PHTML statement without an action or options, then it opens the PHTML destination and creates PHTML output.

Actions

The following table lists the actions available for the ODS PHTML statement. For complete descriptions of these actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.24 ODS PHTML Action Summary Table

Task	Action
Close the PHTML destination and the file that is associated with it	CLOSE
Exclude output objects from the PHTML destination	EXCLUDE
Select output objects for the PHTML destination	SELECT
Write to the SAS log the current selection or exclusion list for the PHTML destination	SHOW

Options

The following table lists the options available for the ODS PHTML statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.25 ODS PHTML Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view the HTML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the HTML output	CHARSET=
Open the PHTML destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the PHTML destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=

Task	Option
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify HTML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the PHTML files that the destination writes to	METATEXT=
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the PHTML destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the PHTML destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=

Task	Option
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS PHTML statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. SAS supplies many markup languages for you to use ranging from DOCBOOK to TROFF. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

ODS PRINTER Statement

Opens, manages, or closes the PRINTER destination, which produces printable output.

Valid: anywhere

Category: ODS: Third-Party Formatted

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly, because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
```

CAUTION:
When you are producing PostScript output, verify that your online viewer or printer is set to use the same paper size as the value that is specified by the OPTIONS PAPERSIZE= statement. Otherwise, some parts of your output might appear to be missing. △

Syntax

ODS PRINTER <(<ID=>*identifier*)> <*action*>;

ODS PRINTER <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS PRINTER statement in the UNIX, VMS, or z/OS operating environments without an action or options, then it opens the PRINTER destination and creates PostScript output, unless otherwise configured by your system administrator.

If you use the ODS PRINTER statement in the Windows operating environment without an action or options, then it prints to the default Windows printer.

Actions

An *action* can be one of the following:

CLOSE
closes the destination and the file that is associated with it. You cannot print the file until you close the destination.

Tip: When an ODS destination is closed, ODS does not send output to that destination. Closing an unneeded destination frees some system resources.

EXCLUDE *exclusion(s)* | ALL | NONE
excludes output objects from the destination.

Default: NONE

Restriction: The destination must be open for this action to take effect.

Main discussion: "ODS EXCLUDE Statement" on page 110

SELECT *selection(s)* | ALL | NONE
selects output objects for the destination.

Default: ALL

Restriction: The destination must be open for this action to take effect.

Main discussion: "ODS SELECT Statement" on page 264

SHOW
writes the current selection or exclusion list for the destination to the SAS log.

Restriction: The destination must be open for this action to take effect.

Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.

See also: "ODS SHOW Statement" on page 277

Options

Table 5.26 ODS PRINTER Option Summary Table

Task	Option
Specify the root name for the anchor tag that identifies each output object in the current file	ANCHOR=
Insert the text string that you specify as the author in the metadata of a file	AUTHOR=
Specify a string to use as the first part of all references that ODS creates in the file	BASE=
Specify whether to generate and display the list of bookmarks for a PDF file	BOOKMARKLIST=
Control the generation of bookmarks in a PDF file	BOOKMARKGEN=
Apply a specified color scheme to your output	COLOR=
Specify the number of columns to create on each page of output	COLUMNS=
Specify the compression of a PDF file. Compression reduces the size of the file	COMPRESS=
Control the generation of a printable table of contents	CONTENTS=

Task	Option
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify the image resolution in dots per inch for output images	DPI=
Specify the file to write to	FILE=
Use the printer drivers that the host system provides	HOST
Open multiple instances of the same destination at the same time	ID=
Insert a string of keywords into the output file's metadata	KEYWORDS=
Create a new file at the specified *starting-point*	NEWFILE=
Omit the table of contents (Bookmark list) that is produced by default when producing PDF or PDFMARK output	NOTOC
Specify that the output from the destination be added to an ODS package	PACKAGE
Create PCL output	PCL
Create PDF output	PDF
Insert special markup which is used when converting a PostScript file to a PDF file	PDFMARK
Control whether notes are added to a PDF file for items that are associated with the FLYOVER= style attribute	PDFNOTE
Control the level of the expansion of the table of contents in PDF documents	PDFTOC=
Create output that is formatted for the specified printer	PRINTER=
Create PostScript output	PS
Control page breaks	STARTPAGE=
Specify the style definition to use in writing the PDF output	STYLE=
Insert the text string that you specify as the subject in the metadata of a file	SUBJECT=
Insert text into your output	TEXT=
Insert the text string that you specify as the title in the metadata of a file	TITLE=
For multi-page tables, provide uniformity from page to page within a single table	UNIFORM

ANCHOR='*anchor-name*'
specifies the root name for the anchor tag that identifies each output object in the current file.

Each output object must have an anchor tag for the bookmarks to reference. The references, which are automatically created by ODS, point to the name of an anchor. Therefore, each anchor name in a file must be unique.

anchor-name
is the root name for the anchor tag that identifies each output object in the current file.

ODS creates unique anchor names by incrementing the name that you specify. For example, if you specify ANCHOR='tabulate', then ODS names the first anchor **tabulate**. The second anchor is named **tabulate1**; the third is named **tabulate2**, and so on.

Requirement: You must enclose *anchor-name* in quotation marks.

Alias: NAMED_DEST= | BOOKMARK=

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Tip: You can change anchor names as often as you want by submitting the ANCHOR= option in a valid statement anywhere in your program. Once you have specified an anchor name, it remains in effect until you specify a new one.

Tip: Specifying new anchor names at various points in your program is useful when you want to link to specific parts of your PRINTER output. Because you can control where the anchor name changes, you know in advance what the anchor name will be at those points.

AUTHOR= '*author-text*'
inserts into the metadata of a file, the text string that you specify as the author.

author-text
is the text in the metadata of an open file that indicates the author.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Restriction: The AUTHOR= option takes effect only if specified at the opening of a file.

Requirement: You must enclose *author-text* in quotation marks.

BASE='*base-text*'
specifies the text to use as the first part of all references that ODS creates in the output file.

base-text
is the text that ODS uses as the first part of all references that ODS creates in the file.

Consider this specification:

```
BASE='http://www.your-company.com/local-url/'
```

In this case, ODS creates references that begin with the string **http://www.*your-company.com/local-url/***. The appropriate *anchor-name* completes the link.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Requirement: You must enclose *base-text* in quotation marks.

BOOKMARKLIST= HIDE | NONE | SHOW
specifies whether to generate and display the list of bookmarks for a PDF file.

Note: The generation of the bookmarks is not affected by the setting of this option. Bookmarks are generated by the BOOKMARKGEN= option. △

HIDE
generates a list of bookmarks for your PDF file. The bookmarks are not automatically displayed when you open the PDF file.

NONE
specifies not to generate a list of bookmarks for your PDF file.

Alias: NO | OFF

Alias: NOBOOKMARKLIST is an alias for BOOKMARKLIST=NONE | NO | OFF.

SHOW
generates a list of bookmarks for your PDF file. The bookmarks are automatically displayed when you open the PDF file.

Alias: YES | ON

Alias: BOOKMARKLIST is an alias for BOOKMARKLIST=SHOW | YES | ON.

Default: SHOW

Restriction: This option can be set only when you first open the destination.

Restriction: This option has an affect only when creating PDF or PDFMARK output.

Interaction: The NOTOC option specifies BOOKMARKLIST= OFF and CONTENTS= OFF.

BOOKMARKGEN | NOBOOKMARKGEN | BOOKMARKGEN=
controls the generation of bookmarks in a PDF file.

BOOKMARKGEN
specifies to generate bookmarks in the PDF file.

BOOKMARKGEN=
controls the generation of bookmarks in a PDF file.

NO
specifies not to generate bookmarks in the PDF file.

Alias: OFF

YES
specifies to generate bookmarks in the PDF file.

Alias: ON

NOBOOKMARKGEN
specifies not to generate bookmarks in the PDF file.

Default: YES or BOOKMARKGEN

Restriction: This option can be set only when you first open the destination.

Interaction: If you set BOOKMARKGEN=NO, then the BOOKMARKLIST option is set to NO also.

COLOR=FULL | GRAY | MONO | NO | YES
applies the specified color scheme to your output.

FULL
creates full color output for both text and graphics.

GRAY
creates gray scale output for both text and graphics.

Alias: GREY

MONO
creates monochromatic output for both text and graphics.

Alias: BW

NO

does not use all the color information that the style definition provides.

Tip: If you specify COLOR=NO, then the destination does this:

- generates black and white output
- creates all text and rules in black
- sets the SAS/GRAPH device to produce SAS/GRAPH output in gray scale
- ignores specifications for a background color from the style definition except for the purposes of determining whether to print rules for the table

YES
uses all the color information that a style definition provides, including background color.

Default: YES

Tip: If you choose color output for a printer that does not support color, then your output might be difficult to read.

Tip: In order to actually print in color, you must also

- use a printer that is capable of printing in color
- use the COLORPRINTING SAS system option. For information about the COLORPRINTING system option, see *SAS Language Reference: Dictionary*.

COLUMNS=*n*
specifies the number of columns to create on each page of output.

n
is the number columns per page.

Default: 1

COMPRESS=*n*
controls the compression of a PDF file. Compression reduces the size of the file.

n
specifies the level of compression. The larger the number, the greater the compression. For example, *n*=0 is completely uncompressed, and *n*=9 is the maximum compression level.

Default: 6

Range: 0–9

Restriction: Use this option only with the ODS PDF statement and the ODS PRINTER statement with the PDF option specified.

Restriction: The COMPRESS= option takes effect only if specified at the opening of a file.

Interaction: The COMPRESS= option overrides the DEFLATION system option. First, the DEFLATION system option checked. Next, the ODS PDF statement COMPRESS= option is checked. If the COMPRESS= option is specified, that value is used regardless of the value specified for the DEFLATION system option. For more information, refer to the DEFLATION option in *SAS Language Reference: Dictionary*.

Interaction: The COMPRESS= option overrides the UPRINTCOMPRESSION option. If COMPRESS= is specified, the UPRINTCOMPRESSION system option is then queried. If the system option is off, it will be turned on for this one PDF statement and the PDF file will be compressed. When compression is complete, the UPRINTCOMPRESSION system option is again enabled for all other files to use. For more information, refer to the UPRINTCOMPRESSION system option in *SAS Language Reference: Dictionary*.

CONTENTS= NO | YES

controls the generation of a printable table of contents.

NO
does not generate a printable table of contents.

Alias: NOCONTENTS is an alias for CONTENTS=NO

YES
generates a printable table of contents.

Alias: CONTENTS is an alias for CONTENTS=YES

CSSSTYLE= '*file-specification*'<(*media-type-1* <..*media-type-10*>)>
specifies a cascading style sheet to apply to your output.

file-specification
specifies a file, fileref, or URL that contains CSS code.
file-specification is one of the following:

"*external-file*"
is the name of the external file.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

"*URL*"
is a URL to an external file.

Requirement: You must enclose *external-file* in quotation marks.

(*media-type-1*<.. *media-type-10*>)
specifies one or more media blocks that corresponds to the type of media that your output will be rendered on. CSS uses media type blocks to specify how a document is to be presented on different media: on the screen, on paper, with a speech synthesizer, with a braille device, and so on.

The media block is added to your output in addition to the CSS code that is not contained in any media blocks. By using the *media-type* suboption, in addition to the general CSS code, you can import the section of a CSS file intended only for a specific media type.

Default: If no *media-type* is specified in your ODS statement, but you do have media types specified in your CSS file, then ODS uses the Screen media type.

Range: You can specify up to ten different media types.

Requirement: You must enclose *media-type* in parentheses.

Requirement: You must specify *media-type* next to the *file-specification* specified by the CSSSTYLE= option.

Tip: If you specify multiple media types, all of the style information in all of the media types is applied to your output. However, if there is duplicate style information in different media blocks, then the styles from the last media block are used.

Requirement: CSS files must be written in the same type of CSS produced by the ODS HTML statement. Only class names are supported, with no IDs and no context based selectors. To view the CSS code that ODS creates, you can specify the STYLESHEET= option, or you can view the source of an HTML file and look

at the code between the <STYLE> </STYLE> tags at the top of the file. For an example of a valid for ODS CSS file, see Example 6 on page 178.

Interaction: If both the STYLE= option and the CSSSTYLE= option are specified on an ODS statement, the option specified last is the option that is used.

Featured in: Example 6 on page 178

DPI=

specifies the image resolution for output files.

Default= 150

Restriction: The DPI= option takes effect only if specified at the opening of a file.

FILE='*external-file*' | *fileref*

specifies the file that contains the output.

external-file

is the name of an external file to write to.

Requirement: You must enclose *external-file* in quotation marks.

fileref

is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

Restriction: The FILE=*fileref* option cannot be used in conjunction with the NEWFILE= option .

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

Default: If you do not specify a file to write to, then ODS writes to the file that is specified by one of two SAS system options:

SYSPRINT=

if you are using the Windows operating environment and do not specify any of the following options: PCL, PDF, PDFMARK, PS, or SAS.

PRINTERPATH=

in all other cases.

If the system option does not specify a file, then ODS writes to the default printer. For more information, see the PRINTER= option on page 228.

Interaction: In an ODS printer family statement that refers to an open ODS PRINTER destination, the FILE= option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

HOST

specifies that ODS use the printer drivers that the host system provides.

Interaction: In an ODS printer family statement that refers to an open ODS PRINTER destination, the HOST option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

(<ID=> *identifier*)

enables you to open multiple instances of the same destination at the same time. Each instance can have different options.

identifier
can be numeric or can be a series of characters that begin with a letter or an underscore. Subsequent characters can include letters, underscores, and numerals.

Restriction: If *identifier* is numeric, it must be a positive integer.

Requirement: The ID= option must be specified immediately after the destination name.

KEYWORDS='*keywords-text*'
inserts into the output file's metadata, a string of keywords . The keywords enable a document management system to do topic-based searches.

keywords-text
is the string of keywords.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Restriction: The KEYWORDS= option takes effect only if specified at the opening of a file.

Requirement: You must enclose *keywords-text* in quotation marks.

NEWFILE= *starting-point*
creates a new file at the specified *starting-point*.

starting-point
is the location in the output where you want to create a new file.

ODS automatically names new files by incrementing the name of the file. In the following example, ODS names the first file **REPORT.PS**. Additional body files are named **REPORT1.PS**, **REPORT2.PS**, and so on.

Example:

```
FILE= 'REPORT.PS'
```

starting-point can be one of the following:

BYGROUP
starts a new file for the results of each BY group.

NONE
writes all output to the file that is currently open.

OUTPUT
starts a new file for each output object. For SAS/GRAPH this means that ODS creates a new file for each SAS/GRAPH output file that the program generates.

Alias: TABLE

PAGE
starts a new file for each page of output. A page break occurs when a procedure explicitly starts a new page (not because the page size was exceeded) or when you start a new procedure.

PROC
starts a body file each time that you start a new procedure.

Default: NONE

Restriction: The NEWFILE= option cannot be used in conjunction with the FILE=*fileref* option.

Restriction: The NEWFILE= option cannot be used if you are sending output to a physical printer.

Tip: If you end the filename with a number, then ODS begins incrementing with that number. In the following example, ODS names the first file **MAY5.PS**. Additional body files are named **MAY6.PS**, **MAY7.PS**, and so on.

Example:

```
FILE= 'MAY5.PS'
```

NOTOC

specifies that ODS omit the table of contents (Bookmark list) that is produced by default when producing PDF or PDFMARK output.

Interaction: The NOTOC option specifies BOOKMARKLIST=OFF and CONTENTS= OFF.

PACKAGE *<package-name>*

specifies that the output from the destination be added to a package.

package-name

specifies the name of a package that was created with the ODS PACKAGE statement. If no name is specified, then the output is added to the unnamed package that was opened last.

See also:
"ODS PACKAGE Statement" on page 198

PCL

creates PCL output.

Restriction: Do not use this option in conjunction with the PDF or PS option.

Interaction: If you use the PCL option in an ODS PRINTER statement that refers to an open ODS PRINTER destination, the option will force ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

PDF

creates PDF output.

Restriction: Do not use this option in conjunction with the PCL or PS options.

Interaction: If you use the PDF option in an ODS PRINTER statement that refers to an open ODS PRINTER destination, the option will force ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

PDFMARK

enables ODS to insert special tags into a PostScript file. When you use software such as Adobe Acrobat (not Adobe Viewer), Acrobat Distiller interprets the tags to create a PDF file that contains the following items:

- bookmarks for each section of the output and for each table.
- references for items that are associated with the URL= style attribute.
- notes for items that are associated with the FLYOVER= style attribute. Notes are optional, and are based on the PDFNOTE option.
- author, keywords, subject, and title in the metadata of a file.

Default: Because using PDFMARK implies PostScript output, SAS automatically uses the PostScript driver that SAS supplies with this option.

Restriction: You cannot use the PRINTER= option with the PDFMARK option.

Requirement: To create a PDF file, you must use specialized software, such as Adobe Acrobat Distiller to convert the marked-up PostScript file into a PDF formatted file.

Interaction: In an ODS printer family statement that refers to an open ODS PRINTER destination, the PDFMARK option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

Tip: Use this option only if you plan to distill the output. Otherwise, it uses excess resources and does not enhance the results.

PDFNOTE | NOPDFNOTE
controls whether notes are added to a PDF file for items that are associated with the FLYOVER= style attribute.

PDFNOTE
adds notes to a PDF file for items that are associated with the FLYOVER= style attribute.

NOPDFNOTE
modifies the behavior of PDFMARK so that notes are not added to the file for items that are associated with the FLYOVER= style attribute.

Default: PDFNOTE

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and ODS PRINTER statement with the PDFMARK option specified.

PDFTOC=*n*
controls the level of the expansion of the table of contents in PDF documents.

n
specifies the level of expansion. For example, PDFTOC=**0** results in a fully expanded table of contents, while PDFTOC=**2** results in a table of contents that is expanded to two levels.

Default: 0

Tip: The PDFTOC= can be set after the file has been opened, but only the last specification for a given file is used.

Featured in: Example 1 on page 212

PRINTER= *printer-name*
creates output that is formatted for the specified printer.

printer-name
is the name of the printer for which you want output formatted.

Alias: PRT

Default: If you do not specify a printer, then ODS formats the printer output for the printer that is specified by one of two SAS system options:

- SYSPRINT= if you are using the Windows operating environment and do not specify any of the following options: PCL, PDFMARK, POSTSCRIPT, PS, or SAS.
- PRINTERPATH= in all other cases.

If the system option does not specify a printer, then ODS writes to the default printer driver as specified in the SAS registry or the Windows registry. In the SAS registry, the default printer is specified in **CORE ▶ PRINTING ▶ Default Printer**

Restriction: *printer-name* must match a subkey in either the SAS registry or the Windows printer registry.

Restriction: You cannot use the PRINTER= option with the PCL, PDF, PDFMARK, or PS options.

Interaction: In an ODS printer family statement that refers to an open ODS PRINTER destination, the PRINTER= option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

Tip: The description of the printer includes its destination and device type. If you are using the SAS printer drivers, then you can find a description of the printer in **CORE ▶ PRINTING ▶ PRINTERS ▶** ***selected-printer*** **▶ PRINTER SETUP ▶ OUTPUT**.

If you are using the Windows operating environment and you do not specify the SAS option in the ODS PRINTER statement, then a description of the printer is located in the Windows registry.

Note: *printer-name* is not necessarily a physical printer. It is a description that tells SAS how to format the output, and where the output is located. For example, it could be a file on a disk. △

Tip: To see a list of available printers for SAS printing, use the REGEDIT command. The printers are listed in the Registry Editor window under **CORE ▶ PRINTING ▶ PRINTERS**.

PS

creates PostScript output.

Alias: POSTSCRIPT

Restriction: Do not use this option in conjunction with the PDF or PCL options.

Interaction: If you use the PS option in an ODS PRINTER statement that refers to an open ODS PRINTER destination, the option will force ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the PRINTER Destination" on page 231.

Tip: Specifying this option is equivalent to specifying both the SAS option and PRINTER= POSTSCRIPT.

STARTPAGE=NEVER | NO | NOW | YES

controls page breaks.

NEVER — specifies not to insert page breaks, even before graphics procedures.

CAUTION:
Each graph normally requires an entire page. The default behavior forces a new page after a graphics procedure. STARTPAGE=NEVER turns off that behavior, so specifying STARTPAGE= NEVER might cause graphics to overprint. △

NO — specifies that no new pages be inserted at the beginning of each procedure, or within certain procedures, even if new pages are requested by the procedure code. A new page will begin only when a page is filled or when you specify STARTPAGE=NOW.

CAUTION:
Each graph normally requires an entire page. The default behavior forces a new page after a graphics procedure, even if you use STARTPAGE=NO. STARTPAGE=NEVER turns off that behavior, so specifying STARTPAGE= NEVER might cause graphics to overprint.
△

Alias: OFF

Tip: When you specify STARTPAGE=NO, system titles and footnotes are still produced only at the top and bottom of each physical page, regardless of the setting of this option. Thus, some system titles and footnotes that you specify might not appear when this option is specified.

NOW — forces the immediate insertion of a new page.

Tip: This option is useful primarily when the current value of the STARTPAGE= option is NO. Otherwise, each new procedure forces a new page automatically.

YES — inserts a new page at the beginning of each procedure, and within certain procedures, as requested by the procedure code.

Alias: ON

Default: YES

STYLE=*style-definition*
specifies the style definition to use in writing the printer output.

Default: If you do not specify a style definition, then ODS uses the style definition that is specified in the SAS registry subkey: **ODS ► DESTINATIONS ► PRINTER**. By default, this value is Printer for the PRINTER, PDF, and PS destinations and MonochromePrinter for the PCL destination.

Main discussion: For a complete discussion of style definitions, see "Working with Styles" on page 538.

See also: For instructions on making your own user-defined style definitions, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.

SUBJECT='*subject-text*'
inserts into the metadata of a file the text string that you specify as the subject.

subject-text
is the text in the metadata of a file that indicates the subject.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Restriction: The SUBJECT= option takes effect only if specified at the opening of a file.

Requirement: You must enclose *subject-text* in quotation marks.

TEXT='*text-string*'
inserts a text string into your output.

text-string
is the text that you want to insert into your output.

Requirement: You must enclose *text-string* in quotation marks.

Tip: If you are submitting more than one procedure step and you do not specify the STARTPAGE=NO option, each procedure will force a new page before the output. Therefore, any text that you specify with TEXT= will be on the same page as the previous procedure.

Featured in: Example 1 on page 114

TITLE='*title-text*'
inserts into the metadata of a file the text string that you specify as the title.

title-text
is the text in the metadata of a file that indicates the title.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Restriction: The TITLE= option takes effect only if specified at the opening of a file.

Requirement: You must enclose *title-text* in quotation marks.

UNIFORM
for multiple page tables, ensures uniformity from page to page within a single table. When the UNIFORM option is in effect, ODS reads the entire table before it starts to print it so that it can determine the column widths that are necessary to accommodate all the data. These column widths are applied to all pages of a multiple page table.

Note: With BY-group processing, SAS writes the results of each BY group to a separate table, so the output might not be uniform across BY groups. △

Default: If you do not specify the UNIFORM option, then ODS prints a table one page at a time. This approach ensures that SAS does not run out of memory while processing very large tables. However, it can also mean that column widths vary from one page to the next.

Tip: The UNIFORM option can cause SAS to run out of memory if you are printing a very large table. If this happens, then you can explicitly set the width of each of the columns in the table, and then print the table one page at a time. To do so, you must edit the table definition that you use. For more information, see "What You Can Do With a Table Template" on page 594.

Details

Opening and Closing the PRINTER Destination You can modify an open PRINTER destination with many ODS PRINTER options. However, any of the following options will automatically close the open destination that is referred to in the ODS PRINTER statement, and will also close any files that are associated with it, and then open a new instance of the destination: FILE=, HOST, PCL, PDF, PDFMARK, PRINTER=, PS, or SAS. If you use one of these options, it is best if you explicitly close the destination yourself.

For example, in the following ODS program, the second ODS PRINTER statement closes the PRINTER destination that is opened by the first ODS PRINTER statement. Therefore, the file **`brickstyle.ps`** will not contain output that is formatted with the **`d3d`** style. However, the second ODS PRINTER statement does not affect the PS destination that is opened by the ODS PS statement. The PS destination is still open and the file **`nostyle.ps`** could be modified.

The ODS PRINTER statement opens the PRINTER destination and creates PostScript output.

```
ods printer ps style=brick file='brickstyle.ps';
proc print data=statepop;
run;
```

The ODS PS statement opens the PS destination and creates PostScript output.

```
ods ps file='nostyle.ps';
proc print data=statepop;
run;
```

The ODS PRINTER statement closes the open PRINTER destination and the files that are associated with it. It then opens a new instance of the PRINTER destination and creates PostScript output.

```
ods printer ps style=d3d file='d3dstyle.ps';
proc print data=statepop;
run;
ods printer ps close;
ods ps close;
```

Printing Output Directly to a Printer Printing output directly to a printer using the ODS PRINTER statement depends on your host operating environment.

Note: To print directly to a printer in the z/OS, UNIX, or VMS operating environment, you can use the FILENAME statement. Specific information about your operating environment is required when using the FILENAME statement. See the SAS documentation for your operating environment before using this statement. Commands are also available in some operating environments that associate a fileref with a file and that break that association.
△

Platform	Method for Sending SAS Output to a Printer
z/OS	Use the FILENAME statement with the SYSOUT= DATA set option specified. You can then print to the fileref. Syntax: **filename *your-fileref* sysout=a dest=*printer-name*;** **ods printer file=*your-fileref*;** Example: **filename local sysout=a dest=chpljj21;** **ods printer file=local;**
UNIX	Use the FILENAME statement with the PIPE command to associate a fileref with your **lpr** print command. Syntax: **filename *your-fileref* pipe 'lpr -P *printer-name*';** **ods printer file=*your-fileref*;** Example: **filename local pipe 'lpr -P chpljj21';** **ods printer file=local;**

Platform	Method for Sending SAS Output to a Printer
VMS	Use the FILENAME statement with the PRINTER device type specified to create a printer fileref that you can print to. Syntax: **filename *your-fileref* printer passall=yes queue=*printer-name*;** **ods printer file=*your-fileref*;** Example: **filename local printer passall=yes queue=chpljj21;** **ods printer file=local;**
Windows	If you want to print to your default printer use this code. Syntax: **ods printer;** If you want to print to a printer that is not the default, then use the PRINTER= option to specify the printer name. Syntax: **ods printer printer=*printer-name*;** Example: **ods printer printer=chpljj21;** *Note:* To find out what printers are available, select **Start ▶ Settings ▶ Printers** from the Taskbar. If a printer is listed there, then you can use it with the ODS PRINTER statement. If the printer name has spaces, then you must put the printer name in quotation marks. △

Using ODS PRINTER with Windows When you use the ODS PRINTER statement in the Windows operating environment, ODS will produce output that is formatted for your default Windows printer unless you specify a different printer by using the PRINTER= option on page 228. You can also produce printable output files in PCL, PDF, or PostScript format by using the appropriate option.

Using ODS PRINTER with All Other Hosts When you use the ODS PRINTER statement in any other operating environment, ODS uses the SAS drivers to produce output files in PCL, PDF, or PostScript formats. By default, the ODS PRINTER statement produces PostScript output files. You can also produce printable output files in PCL or PDF format by using the appropriate option or registry setting.

PDF Security In SAS 9.2, you can easily encrypt and password-protect your PDF output files. Two levels of security are available: 40-bit (low) and 128-bit (high). With either of these settings, a password will be required to open a PDF file that has been generated with ODS.

To enable encryption and password protection, specify the OPTIONS statement. The following code shows how to encrypt your PDF output file with a low level of encryption. The PDF file generated will be password protected.

```
options pdfsecurity=low pdfpw=(open=testpw);
```

The following code shows how to encrypt your PDF output file with a high level of encryption that is password protected:

```
options pdfsecurity=high pdfpw=(open=testpw);
```

The following code shows the PDF security option used with the PDF destination:

```
options pdfsecurity=high pdfpw=(open=testpw);
ods pdf file="secure.pdf";
proc contents data=sashelp.class;
 run;
ods pdf close;
```

For detailed information on the PDF Security options, see “Securing ODS Generated PDF Files” on page 36.

Note: Encryption requires Acrobat version 5.0 or later. △

PDF Views Is SAS 9.2, Two new system options enable you to control the way you view your PDF document. The PDFPAGELAYOUT system option controls the page layout. This setting is equivalent to selecting **View ▶ Page Display** in Adobe Acrobat Reader when a document is open. The PDFPAGEVIEW system option controls the page viewing mode. This setting is equivalent to selecting **View ▶ Zoom** in Adobe Acrobat Reader.

Refer to *SAS Language Reference: Dictionary* for detailed information on these system options.

Example

Example 1: Selecting Output for the HTML and PRINTER Destinations

ODS features:

ODS _ALL_ CLOSE

ODS HTML statement:

BODY=

ODS PRINTER statement:

FILE=

PS

ODS LISTING statement:

CLOSE

ODS SELECT statement:

with label

with name

with path

Other SAS features:

PROC UNIVARIATE

Data set:

See “Creating the StatePop Data Set” on page 881.

This example selects three output objects from a UNIVARIATE procedure step to send to both the HTML destination and to the PRINTER destination.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you

might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program

Prevent listing output from being created. The ODS LISTING statement closes the LISTING destination in order to conserve resources.

```
ods listing close;
```

Set the SAS system options. The OPTIONS statement controls several aspects of the PRINTER output. The NODATE system option specifies that SAS not print the date and the time. The NONUMBER system option specifies that SAS not print the page number on the first title line of each page of SAS output. These options do not affect the HTML output.

```
options nodate nonumber;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. BODY= sends all output objects to the external file that you specify. Some browsers require an extension of HTM or HTML on the filename.

```
ods html body='your_file.html';
```

Create PostScript output. The ODS PRINTER statement opens the PRINTER destination and the PS option specifies PostScript output. FILE= sends all output objects to the external file that you specify.

```
 ods printer ps file='your_file.ps';
```

Specify the output objects to send to the open destinations. The ODS SELECT statement specifies three output objects to send to all open destinations. The first output object is selected by its name, **BasicMeasures**. The second output object is selected by its label, **Tests For Location**. These two selection criteria select the output objects for the analysis of both variables. The third output object is selected by its full path **Univariate.CityPop_90.ExtremeObs**. This selection criterion selects the output object for only one variable, CityPop_90.

```
ods select BasicMeasures
           'Tests For Location'
           Univariate.CityPop_90.ExtremeObs;
```

Compute descriptive statistics for two variables. PROC UNIVARIATE computes descriptive statistics for two variables, CityPop_80 and CityPop_90. ODS routes the selected output objects to the HTML and PRINTER destinations.

```
proc univariate data=statepop mu0=3.5;
   var citypop_90 citypop_80;
run;
```

Close the open destinations so that you can view or print the output. The ODS _ALL_ CLOSE statement closes all of the open destinations and all of the files that are associated with them. You must close the destinations before you can view the output with a browser, or before you can send the output to a physical printer.

```
ODS _all_ close;
```

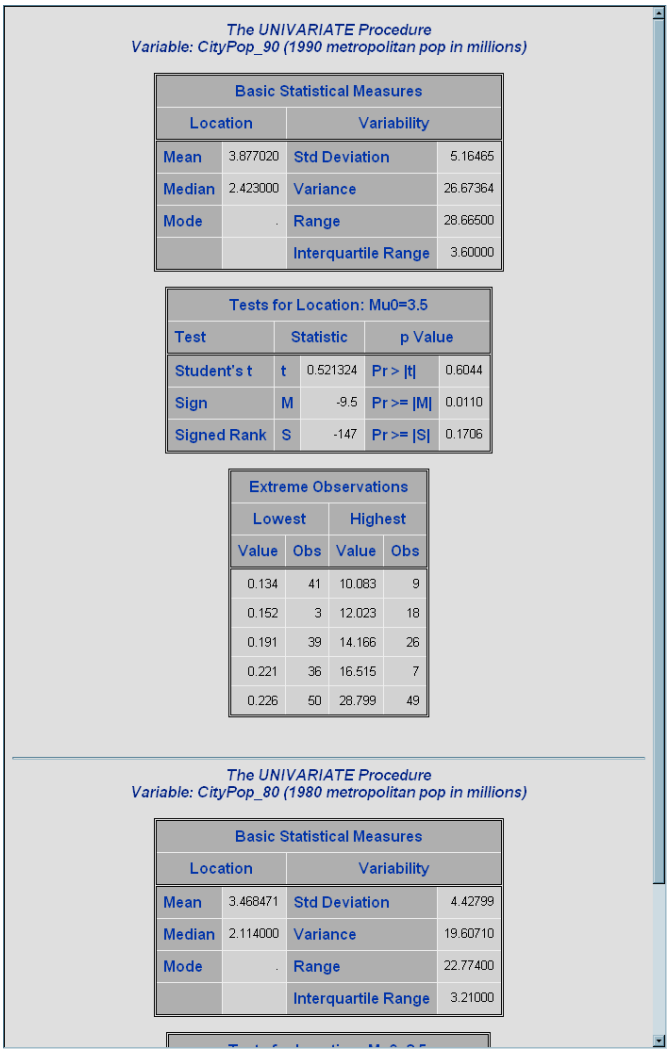
Reset the default output type to LISTING. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods listing;
```

HTML Output

Display 5.21 HTML Output for the Variables CityPop_90 and CityPop_80

The HTML output includes three output objects for the variable CityPop_90, and two output objects for the variable CityPop_80.



The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

Basic Statistical Measures			
Location		Variability	
Mean	3.877020	Std Deviation	5.16465
Median	2.423000	Variance	26.67364
Mode	.	Range	28.66500
		Interquartile Range	3.60000

Tests for Location: Mu0=3.5				
Test		Statistic	p Value	
Student's t	t	0.521324	Pr > |t|	0.6044
Sign	M	-9.5	Pr >= |M|	0.0110
Signed Rank	S	-147	Pr >= |S|	0.1706

Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
0.134	41	10.083	9
0.152	3	12.023	18
0.191	39	14.166	26
0.221	36	16.515	7
0.226	50	28.799	49

The UNIVARIATE Procedure
Variable: CityPop_80 (1980 metropolitan pop in millions)

Basic Statistical Measures			
Location		Variability	
Mean	3.468471	Std Deviation	4.42799
Median	2.114000	Variance	19.60710
Mode	.	Range	22.77400
		Interquartile Range	3.21000

Printer Output

Display 5.22 Partial PostScript Output for the Variables CityPop_90 and CityPop_80

The printer output includes three output objects for the variable CityPop_90, and two output objects for the variable CityPop_80.

The SAS System

The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

Basic Statistical Measures			
Location		**Variability**	
Mean	3.877020	**Std Deviation**	5.16465
Median	2.423000	**Variance**	26.67364
Mode	.	**Range**	28.66500
		Interquartile Range	3.60000

Testsfor Location:Mu0=3.5				
Test	**Statistic**		**p Value**	
Student's t	**t**	0.521324	**Pr > |t|**	0.6044
Sign	**M**	-9.5	**Pr >= |M|**	0.0110
Signed Rank	**S**	-147	**Pr >= |S|**	0.1706

Extreme Observations			
Lowest		**Highest**	
Value	**Obs**	**Value**	**Obs**
0.134	41	10.083	9
0.152	3	12.023	18
0.191	39	14.166	26
0.221	36	16.515	7
0.226	50	28.799	49

ODS PROCLABEL Statement

Enables you to change a procedure label.

Valid: anywhere

Category: ODS: Output Control

Interaction: This statement applies to all open destinations, except for the output destination, where a procedure label is not an option. However, this setting lasts for only one procedure step. You must issue an ODS PROCLABEL statement for each procedure step that you have.

Syntax

ODS PROCLABEL '*string*';

Required Arguments

'string'
is the procedure label that you specify.

Interaction: The NOLABEL system option overrides the ODS PROCLABEL statement. Therefore, to produce labels using the ODS PROCLABEL statement, you must specify the LABEL system option also.

Details

ODS PROCLABEL affects the item names in the outer list of the table of contents.

See Also

System Option:
"Label System Option" in *SAS Language Reference: Dictionary*

ODS PROCTITLE Statement

Determines whether to write the title that identifies the procedure that produces the results in the output.

Valid: anywhere

Category: ODS: Output Control

Interaction: This statement applies to all open destinations, except for the output destination, where a procedure label is not an option. This setting persists until you issue an ODS NOPROCTITLE statement. You do not have to issue an ODS PROCTITLE statement for each procedure step.

Syntax

ODS PROCTITLE | NOPROCTITLE;

Required Arguments

ODS PROCTITLE
writes, in the output, the name of the procedure that produces the results.

Note: Not all procedures use a procedure title. △

Default: ODS PROCTITLE is the default.

ODS NOPROCTITLE
suppresses the writing of the title of the procedure that produces the results.

Details

The following table lists the aliases for the ODS PROCTITLE statement:

Statement	Alias
ODS PROCTITLE	ODS PROCTITLE=ON
	ODS PTITLE
	ODS PTITLE=ON
	ODS PTITLE=YES
ODS NOPROCTITLE	ODS PROCTITLE=OFF
	ODS NOPTITLE
	ODS PTITLE=OFF
	ODS PTITLE=NO

ODS PS Statement

Opens, manages, or closes the PS destination, which produces PostScript (PS) output.

Valid: anywhere

Category: ODS: Third-Party Formatted

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly, because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
```

CAUTION:
When you are producing PostScript output, verify that your online viewer or printer is set to use the same paper size as the value that is specified by the OPTIONS PAPERSIZE= statement. Otherwise, some parts of your output might appear to be missing. △

Syntax

ODS PS <(<ID=>*identifier*)> <*action*>;

ODS PS <(<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS PS statement without an action or options, then it opens the PS destination and creates PostScript output.

Actions

The following table lists the actions that are available for the ODS PS statement. For complete descriptions of actions see “Actions” on page 219 in the ODS PRINTER statement.

Table 5.27 ODS PS Action Summary Table

Task	Action
Close the PCL destination and the file that is associated with it	CLOSE
Exclude output objects from the PCL destination	EXCLUDE
Select output objects for the PCL destination	SELECT
Write to the SAS log the current selection or exclusion list for the PCL destination	SHOW

Options

The following table lists the options available for the ODS PS statement. For more detailed descriptions of these options, see “Options” on page 219 in the ODS PRINTER statement.

Table 5.28 ODS PS Option Summary Table

Task	Option
Specify the root name for the anchor tag that identifies each output object in the current file	ANCHOR=
Insert the text string that you specify as the author in the metadata of a file	AUTHOR=
Specify a string to use as the first part of all references that ODS creates in the file	BASE=
Specify whether to generate and display the list of bookmarks for a PS file	BOOKMARKLIST=
Control the generation of bookmarks in a PS file	BOOKMARKGEN=
Apply a specified color scheme to your output	COLOR=
Specify the number of columns to create on each page of output	COLUMNS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify the image resolution for output files	DPI=
Specify the file to write to	FILE=

Task	Option
Open multiple instances of the same destination at the same time	ID=
Insert a string of keywords into the output file's metadata	KEYWORDS=
Create a new file at the specified *starting-point*	NEWFILE=
Specify that the output from the destination be added to an ODS package	PACKAGE
Insert special markup which is used when converting a PostScript file to a PDF file	PDFMARK
Control whether notes are added to a PDF file for items that are associated with the FLYOVER= style attribute	PDFNOTE
Control page breaks	STARTPAGE=
Specify the style definition to use in writing the PS output	STYLE=
Insert text into your output	TEXT=
For multipage tables, ensure uniformity from page to page within a single table	UNIFORM

Details

The ODS PS statement is part of the ODS printer family of statements. Statements in the printer family open the PCL, PDF, PRINTER, or PS destination, producing output that is suitable for a high-resolution printer. The ODS PCL, ODS PDF, and ODS PRINTER statements are also members of the ODS printer family of statements.

Opening and Closing the PS Destination You can modify an open PS destination with many ODS PS options. However, the FILE=, PDFMARK, and SAS options will automatically close the open destination that is referred to in the ODS PS statement and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

See Also

Statements:

"ODS PCL Statement" on page 208
"ODS PDF Statement" on page 210
"ODS PRINTER Statement" on page 218
"The Third-Party Formatted Destinations" on page 26

ODS RESULTS Statement

Tracks ODS output in the Results window.

Valid: anywhere

Category: ODS: Output Control

Restriction: Valid in a windowing environment only, not in batch mode.

Alias: ODS RESULTS|NORESULTS;

Syntax

ODS RESULTS ON | OFF;

Required Arguments

ON
tracks output that ODS generates in the Results window.

OFF
turns off the tracking of output that ODS generates in the Results window.

Details

Using ODS RESULTS ON sends all output to the Results window. This is the default setting. Using ODS RESULTS OFF disables ODS tracking, and output is not sent to the Results window. The OFF option is recommended for long-running jobs, such as regression analyses, when you do not want to track all of the output.

ODS RTF Statement

Opens, manages, or closes the RTF destination, which produces output written in Rich Text Format for use with Microsoft Word 2002.

Valid: anywhere

Category: ODS: Third-Party Formatted

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment that does not have SAS software installed, this output will not be displayed correctly. The SAS Monospace font is not recognized if SAS is not installed. For the correct display of your document, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
```

Interaction: To change the page orientation of the RTF output, specify the system option ORIENTATION=. To change the orientation, you will need to trigger the change by issuing the ODS RTF statement after the global options statement. See Example 3 on page 261 for details.

Tip: Microsoft Word 2002 is the current, official, minimum level that is supported. However, no problems have been found with Microsoft Word 2000 and SAS RTF files.

Tip: When producing large tables, use the ODS TAGSETS.RTF statement. For detailed information, see “ODS TAGSETS.RTF Statement” on page 286.

Syntax

ODS RTF <(<ID=> *identifier*)> *action*;

ODS RTF <(<ID=> *identifier*)> <*option(s)*>;

Actions

The following table lists the actions that are available for the ODS RTF statement.

Table 5.29 ODS RTF Action Summary Table

Task	Action
Close the RTF destination and the file that is associated with it	CLOSE
Exclude output objects from the RTF destination	EXCLUDE
Select output objects for the RTF destination	SELECT
Write to the SAS log the current selection or exclusion list for the RTF destination	SHOW

CLOSE
closes the RTF destination and any files that are associated with it.

Tip: When you close an ODS destination, ODS does not send output to that destination. To free more system resources, close destinations that you do not need.

EXCLUDE *exclusion(s)* | ALL | NONE
excludes output objects from the RTF destination.

Restriction: The destination must be open for this action to take effect.

Default: NONE

See also: "ODS EXCLUDE Statement" on page 110

SELECT *selection(s)* | ALL | NONE
selects output objects for the RTF destination.

Default: ALL

Restriction: The destination must be open for this action to take effect.

See also: "ODS SELECT Statement" on page 264

SHOW
writes the current selection list or exclusion list for the destination to the SAS log.

Restriction: The destination must be open for this action to take effect.

See also: "ODS SHOW Statement" on page 277

Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.

Options

The following table lists the options that are available for the ODS RTF statement.

Table 5.30 ODS RTF Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify the text string that identifies the author. This text string is inserted into the metadata of a file.	AUTHOR=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Specify that the titles and footnotes are to be placed into the body of the RTF document and not into the header and footer sections	BODYTITLE
Specify that the titles and footnotes are to be placed into the body of the RTF document and not into the header and footer sections. The titles and footnotes will also be placed into cells or tables.	BODYTITLE_AUX
Specify the number of columns to create on each page of output	COLUMNS=
Specify whether to produce a table of contents page	CONTENTS
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify a device for the RTF output destination	DEVICE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Open the ODS RTF destination and specify the name of the file to which to write information	FILE=
Specify the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Open multiple instances of the same destination at the same time	ID=
Specify the image resolution for the graphical output	IMAGE_DPI=
Control where tables split on a page	KEEPN | NOKEEPN
Create a new body file at the specified starting point	NEWFILE=
Suppress currently defined footnotes in the graphics file. They appear in the RTF file instead.	NOGFOOTNOTE
Suppress currently defined titles in the graphics file. They appear in the RTF file instead.	NOGTITLE
Specify whether contents data is inserted into the RTF file	NOTOC_DATA |TOCDATA
Insert the text that you specify into the metadata of the RTF file	OPERATOR=
Specify that the output from the destination be added to an ODS package	PACKAGE

Task	Option
Specify the location of an aggregate storage location or a SAS catalog for all RTF files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Write to the RTF file the time and date that you started your SAS session	SASDATE
Control page breaks	STARTPAGE=
Specify a style definition to use in writing the RTF files	STYLE=
Insert text into your RTF output	TEXT=
Insert the text string that you want as your title into the metadata of a file	TITLE=
Specify a translation table to use when you transcode a file for output	TRANTAB=

ANCHOR= '*anchor-name*'
specifies the base name for the RTF anchor tag that identifies each output object in the current file.

Each output object must have an anchor tag to which other files will link or reference. The references, which ODS automatically creates, point to the name of an anchor. Therefore, each anchor name in a file must be unique.

anchor-name
is the base name for the RTF anchor tag that identifies each output object in the current file.

ODS increments the name that you specify and creates unique anchor names. For example, if you specify ANCHOR= 'tabulate', then ODS names the first anchor **`tabulate`**. The second anchor is named **`tabulate1`**; the third is named **`tabulate2`**, and so on.

Requirement: You must enclose *anchor-name* in quotation marks.

Alias: NAMED_DEST= | BOOKMARK=

Tip: It is useful to specify new anchor names at various points in your program when you want other RTF files to link to specific parts of your RTF output. Because you can control where the anchor name changes, you know in advance what the anchor name will be at those points.

Tip: You can change anchor names as often as you want by submitting the ANCHOR= option in an ODS RTF statement anywhere in your program. After you specify an anchor name, it remains in effect until you specify a new one.

AUTHOR= '*author-text*'
inserts the text string that you specify as the author into the metadata of a file.

author-text
is the text in the metadata of an open file that indicates the author.

Requirement: You must enclose *author-text* in quotation marks.

BASE= '*base-text*'
specifies the text to use as the first part of references that ODS creates in the output file.

base-text

is the text that ODS uses as the first part of all references that ODS creates in the file.

Consider this specification:

```
BASE='http://www.your-company.com/local-url/'
```

In this case, ODS creates links that begin with the string **http://www.your-company.com/local-url/**.

Requirement: You must enclose *base-text* in quotation marks.

BODYTITLE

specifies that SAS titles and footnotes are placed into the body of the RTF document instead of into the headers and footers section of the RTF document.

Restriction: The BODYTITLE option can be specified only when you create a new RTF file.

Interaction: When you set the STARTPAGE= option to YES (the default), ODS inserts a new page at the start of each procedure. ODS relies on Word to place headers and footers correctly before and after the procedures. When you specify BODYTITLE, titles and footnotes are removed from the header and footer sections of the RTF document. Titles and footnotes are then placed into the body of the document, and are appended to every TABLE. Therefore, when you set the STARTPAGE= option to YES and specify the BODYTITLE option, the titles and footnotes might not repeat on every page. For example, if there is a table that spans multiple pages, the title will be on the first page only, and the footnote will be on the last page only.

Note: When you specify the BODYTITLE option, Microsoft Word no longer controls the placement of the header and footer text, but Microsoft Word still controls other header and footer information, such as page number and date. △

Tip: The background is not honored on the title cells.

See also: BODYTITLE_AUX option. Use the BODYTITLE_AUX option when you want titles and footnotes placed in tables in the body of the RTF document.

BODYTITLE_AUX

specifies that SAS titles and footnotes be placed into the body of the RTF document instead of into the headers and footers section of the RTF document. These titles and footnotes are put into cells, which allows titles and footnotes to be centered, left-justified, or right-justified.

Restriction: You can specify the BODYTITLE_AUX option only when you are creating a new RTF file.

Interaction: When you set the STARTPAGE= option to YES (the default), ODS inserts a new page at the start of each procedure and relies on Word for the correct placement of headers and footers before and after the procedures. When you specify BODYTITLE_AUX, titles and footnotes are removed from the header and footer sections of the RTF document. Titles and footnotes are then placed into the body of the document, and they are appended to every TABLE. Therefore, when you set the STARTPAGE= option to YES and you specify the BODYTITLE_AUX option, the titles and footnotes might not repeat on every page. For example, if there is a table that spans multiple pages, then the title will be on the first page only, and the footnote will be on the last page only.

Note: When you specify the BODYTITLE_AUX option, Microsoft Word no longer controls the placement of the header and footer text, but Microsoft Word still controls other header and footer information, such as page number and date. △

See also: BODYTITLEoption

Featured in: Example 2 on page 259

COLUMNS= *n* | MAX
specifies the number of columns to place across each page of output.

n
is the number of one-inch columns that you want on the page.

MAX
specifies the maximum number of one-inch-wide columns for the paper size and margin setting. This value is dependent upon the paper size and page orientation.

Default: the number of one-inch columns that fit on the page.

Tip: Titles are considered tables and not RTF instructions in Measured RTF (ODS TAGSETS.RTF statement). When you use the COLUMNS= option with Measured RTF, titles will appear at the top of each column. However, ODS truncates the titles to fit the column width.

Tip: If you specify a value greater than the maximum number of one inch columns that can fit on the page, a note is printed to the SAS log that states what the maximum value can be for that page.

Interaction: When you specify the COLUMNS= option, the STARTPAGE=NO option will not be honored.

CONTENTS
produces a table of contents page for RTF documents that are opened in Microsoft Word. The table of contents page contains a Table of Contents field, which puts all of the contents information that is embedded in the document into a table of contents. To expand the table of contents, right-click under the title in Microsoft Word and select **`Update Field`** from the selection list.

Restriction: Do not use the CONTENTS option with the NEWFILE option.

Tip: To go to a specific topic in the document, you can double-click or hold down the CTRL key and click on the topic in the table of contents. You might have to configure Microsoft Word to use the CTRL-click method by selecting **Tools** ▶ **Options** ▶ **Edit** and checking **`Use CTRL + Click to follow hyperlink`**.

Tip: You must specify the TOC_DATA option to view the text that is captured in the Table of Contents. If not, the Table of Contents page displays the error message "Error! No table of contents entries found." NOTOC_DATA is the default option that is used.

See also: TOC_DATA option

Featured in: Example 1 on page 256

CSSSTYLE= *'file-specification'*<(*media-type-1 <…media-type-10>*)>
specifies a cascading style sheet to apply to your output.

file-specification
specifies a file, fileref, or URL that contains CSS code.
file-specification is one of the following:

"*external-file*"
is the name of the external file.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

"URL"
is a URL to an external file.

Requirement: You must enclose *external-file* in quotation marks.

(*media-type-1*<… *media-type-10*>)
specifies one or more media blocks that correspond to the type of media to which your output will be rendered. CSS uses media type blocks to specify the different media on which a document is to be presented: on the screen, on paper, with a speech synthesizer, with a braille device, and so on.

ODS adds the media block to your output in addition to the CSS code that is not contained in any media blocks. By using the *media-type* suboption and the general CSS code, you can import the section of a CSS file intended only for a specific media type.

Default: If you do not specify a *media-type* in your ODS statement, but you do specify media types in your CSS file, ODS uses the Screen media type.

Range: You can specify up to ten different media types.

Requirement: You must enclose *media-type* in parentheses.

Requirement: You must specify *media-type* next to the *file-specification* specified by the CSSSTYLE= option.

Tip: If you specify multiple media types, ODS applies all of the style information in all of the media types to your output. However, if there is duplicate style information in different media blocks, then ODS uses the styles from the last media block.

Interaction: If you specify both the STYLE= option and the CSSSTYLE= option on an ODS statement, ODS uses the last option that you specified.

Requirement: CSS files must be written in the same type of CSS produced by the ODS HTML statement. Only class names are supported, with no IDs and no context-based selectors. To view the CSS code that ODS creates, you can specify the STYLESHEET= option, or you can view the source of an HTML file and look at the code between the <STYLE> </STYLE> tags at the top of the file. For an example of a valid ODS CSS file, see Example 6 on page 178.

Featured in: Example 6 on page 178

DEVICE= ***device-driver***
specifies the name of a device driver. ODS automatically selects an optimal default device for each open output destination.

The following table lists default devices for the most common ODS output destinations.

Output Destination	Default Device
HTML	PNG
LISTING	Host Specific Display Device (PC- WIN, UNIX - XColor, MVS -Display Device)
Measured RTF	PNG
RTF	SASEMF
PCL	SASPRTM (Monochrome Output)*
PDF	SASPRTC (Color Output)*
POSTSCRIPT	SASPRTC (Color Output)*
PRINTER	Host Specific Default Printer*

* Does not support changing the default device in the SAS Registry.

Tip: Specifying a device on the ODS DEVICE= option takes precedence over the SAS global option and the graphics option.

See: "DEVICE= System Option" in *SAS Language Reference: Dictionary*. Also refer to "Device Drivers" in *SAS/GRAPH: Reference* for information on selecting device drivers.

ENCODING= ***local-character-set-encoding***
overrides the encoding for input or output processing (transcodes) of external files.

See: For information about the ENCODING= option, see *SAS National Language Support (NLS): Reference Guide*.

FILE= ***'external-file' | fileref***
opens the RTF destination and specifies the RTF file or SAS catalog to which to write. This file remains open until you do one of the following actions:

- Close the RTF destination with ODS RTF CLOSE or ODS _ALL_ CLOSE.
- Specify a different file to which to write.

external-file
is the name of an external file to which to write.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See also: The section on statements in *SAS Language Reference: Dictionary* for information about the FILENAME statement.

Restriction: You cannot use the FILE=*fileref* option with the NEWFILE= option.

Alias: BODY=

Interaction: In an ODS RTF statement that refers to an open RTF destination, the FILE= option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see "Opening and Closing the RTF Destination" on page 255.

See also: NEWFILE=

GFOOTNOTE | NOGFOOTNOTE
controls the location of the footnotes that are defined by the graphics program that generates the RTF output.

GFOOTNOTE
includes all of the currently defined footnotes within the graphics output.

NOGFOOTNOTE
prevents all of the currently defined footnotes from appearing in the graphics file. Instead, they become part of the RTF file.

Default: GFOOTNOTE

Restriction: This option applies only to SAS programs that produce one or more device-based graphics, or graphics created by the SGPLOT procedure, the SGPANEL procedure, or the SGSCATTER procedure.

GTITLE | NOGTITLE
controls the location of the titles that are defined by the graphics program that generates the RTF output.

GTITLE

includes all of the currently defined titles within the graphics output that is called by the body file.

NOGTITLE
prevents all of the currently defined titles from appearing in the graphics output. Instead, the titles become part of the RTF file.

Default: GTITLE

Restriction: This option applies only to SAS programs that produce one or more device-based graphics, or graphics created by the SGPLOT procedure, the SGPANEL procedure, or the SGSCATTER procedure.

(ID= *identifier*)

identifier
can be a number or a series of characters that begin with a letter or an underscore.

Restriction: If *identifier* is a number, the number must be positive.

Requirement: You must specify the ID= option immediately after the destination name.

Tip: You can omit the ID= option, and use a name or a number instead to identify the instance.

Featured in: Example 1 on page 212

IMAGE_DPI
specifies the image resolution for graphical output.

Default: 200

Restriction: The IMAGE_DPI= option affects template-based graphics only.

KEEPN | NOKEEPN
controls where tables split on a page.

KEEPN
ODS allows table splits only if the entire table cannot fit on one page.

NOKEEPN
ODS lets a table split at a page break.

Tip: Although KEEPN minimizes page breaks in tables, it might use substantially more paper than NOKEEPN because the KEEPN option issues a page break before starting to print any table that does not fit on the remainder of the page.

NEWFILE= *starting-point*
creates a new file at the specified *starting-point*.
starting-point can be one of the following:

BYGROUP
starts a new file for the results of each BY group.

NONE
writes all output to the next file that you open, and then stops incrementing.

OUTPUT
starts a new file for the results of each BY group.

Alias: TABLE

PROC
starts a new file each time that you start a new procedure.

Default: NONE

Restriction: You cannot use both the NEWFILE= and TEXT= options in the same ODS RTF statement. You must use a separate ODS RTF statement for each of these options.

Restriction: You cannot use the NEWFILE= option with the FILE=*fileref* option.

Tip: If you end the filename with a number, then ODS begins incrementing with that number. In the following example, ODS names the first body file MAY5.XML, and names additional body files MAY6.XML, MAY7.XML, and so on.

NOGFOOTNOTE
See the description of GFOOTNOTE | NOFOOTNOTE in this section.

NOGTITLE
See the description of GTITLE | NOGTITLE in this section.

NOTOC_DATA
See the description of TOC_DATA in this section.

OPERATOR= *'text-string'*
inserts the text you specify into the metadata of the RTF file.

text-string
is the text in the metadata of a file that indicates the author.

Requirement: You must enclose *text-string* in quotation marks.

PACKAGE *<package-name>*
specifies that the output from the destination be added to a package.

package-name
specifies the name of a package that was created with the ODS PACKAGE statement. If no name is specified, then the output is added to the unnamed package that was opened last.

See also: "ODS PACKAGE Statement" on page 198

PATH= *'aggregate-file-storage-specification'* | *fileref* | *libref.catalog* (URL= *'Uniform-Resource-Locator'* | NONE)
specifies the location of an aggregate storage location or a SAS catalog for all RTF files. If the GPATH= option is not specified, all graphics output files are written to the *"aggregate-file-storage-specification"* or *libref*.

'*aggregate-file-storage-location*'
specifies an aggregate storage location such as directory, folder, or partitioned data set.

Requirement: You must enclose *aggregate-file-storage-location* in quotation marks.

fileref
is a file reference that has been assigned to an aggregate storage location. Use the FILENAME statement to assign a fileref.

Interaction: If you use a fileref in the PATH= option, then ODS does not use information from PATH= when it constructs links.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

libref.catalog
specifies a SAS catalog to write to.

See: For information about the LIBNAME statement, see *SAS Language Reference: Dictionary*.

URL= '*Uniform-Resource-Locator*' | NONE
specifies a URL for the *file-specification*.

Uniform-Resource-Locator
is the URL you specify. ODS uses this URL instead of the filename in all the links and references that it creates to the file.

NONE
specifies that no information from the PATH= option appears in the links or references.

Tip: This option is useful for building output files that can be moved from one location to another. The links from the contents and page files must be constructed with a single-name URL, and the contents, page, and body files must be in the same location.

Interaction: If you use the BODY= or FILE= external file option in conjunction with the PATH= option, the external file specification should not include path information.

RECORD_SEPARATOR= '*alternative-separator*' | NONE
specifies an alternative record separator, which is a character or string that separates lines in the output files.

Different operating environments use different separator characters. If you do not specify a record separator, ODS formats the RTF files for the environment in which you run the SAS job. However, if you are generating files in one operating environment to view in another operating environment that uses a different separator character, you can specify a record separator that is appropriate for the target environment.

alternative-separator
represents one or more characters in hexadecimal or ASCII format. For example, the following option specifies a record separator of a carriage-return character and a linefeed character (on an ASCII file system):

```
RECORD_SEPARATOR= '0D0A'x
```

Operating Environment Information: In a mainframe environment, the option that specifies a record separator for a carriage-return character and a linefeed character for use with an ASCII file system is:

```
RECORD_SEPARATOR= '0D25'x
```

△

Requirement: You must enclose *alternative-separator* in quotation marks.

NONE
produces RTF output that is appropriate for the environment in which you run the SAS job.

Operating Environment Information: In many operating environments, using a value of NONE has the same result as omitting the RECORD_SEPARATOR option. △

Operating Environment Information: In a mainframe environment, by default, ODS produces a binary file that contains embedded record-separator characters. This approach means that the file is not restricted by the line-length restrictions on ASCII files, but it also means that the lines are concatenated if you view the file in an editor.

If you want to format the RTF files in a manner that allows you to read them with an editor, use RECORD_SEPARATOR= NONE. In this case, ODS writes one

line of RTF at a time to the file. When you use a value of NONE, the logical record length of the file to which you are writing must be at least as long as the longest line that ODS produces. Otherwise, RTF might wrap to another line at an inappropriate place. △

Alias:

RECSEP=

RS=

SASDATE

writes to the RTF file the time and the date that you started your SAS session.

Restriction: You can specify SASDATE only when you open a new file. If you specify the option at any other time, ODS writes a warning message to the SAS log.

Interaction: To reset the SAS session time that is input into the RTF file, use the DTRESET system option.

See: For information about the DTRESET system option, see *SAS Language Reference: Dictionary*.

STARTPAGE= YES | NO | NOW

controls page breaks.

YES

inserts a new page at the start of each procedure and within certain procedures, as is requested by the procedure code.

Alias: ON

Interaction: When the STARTPAGE= option is set to YES (the default), ODS inserts a new page at the start of each procedure and relies on Word for the correct placement of headers and footers before and after the procedures. When you specify BODYTITLE, titles and footnotes are removed from the header and footer sections of the RTF document. Titles and footnotes are then placed into the body of the document, and they are appended to every TABLE. Therefore, when you set the STARTPAGE= option to YES and you specify the BODYTITLE option, the titles and footnotes might not repeat on every page. For example, if there is a table that spans multiple pages, the title will appear on only the first page, and the footnote will appear on only the last page.

Note: When you specify the BODYTITLE= option, Microsoft Word no longer controls the placement of the headers and footers text, but it still controls other header and footer information, such as page number and date. △

NO

instructs ODS not to insert any new pages at the start of each procedure or within certain procedures, even if the procedure code requests new pages. A new page begins only when a page is filled or when you specify STARTPAGE= NOW.

Alias: NEVER

Tip: This option prints only the first set of titles and the first set of footnotes to the RTF file.

Interaction: When you specify the COLUMNS= option, the STARTPAGE=NO option is not honored.

NOW

forces the immediate insertion of a new page.

Tip: This option is useful primarily when the current value of the STARTPAGE= option is NO. Otherwise, each new procedure forces a new page automatically.

Tip: Specifying STARTPAGE= NO prevents forced page breaks. You can turn on forced page breaking again by specifying STARTPAGE=YES. You can insert a page break at any time by specifying STARTPAGE=NOW.

Default: YES

STYLE= *style-definition*
specifies the style definition for ODS to use to write the RTF files.

style-definition
describes how to display the presentation aspects (color, font face, font size, and so on) of your SAS output. A style definition determines the overall appearance of the documents that use that style definition. Each style definition consists of style elements.

Main discussion: For a complete discussion of style definitions, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.

Default: If you do not specify a style definition, ODS uses the file that is specified in the SAS registry subkey: **ODS ▶ DESTINATIONS ▶ RTF**. By default, this value specifies **RTF** for traditional RTF and Measured RTF.

TEXT= '*text-string*'
inserts text into your RTF output.

text-string
is the text that you want to insert into your RTF output. You can also use TEXT= to annotate other output.

Restriction: You cannot use both the NEWFILE= and TEXT= options in the same ODS RTF statement. You must use a separate ODS RTF statement for each of these options.

Requirement: You must enclose a *text-string* in quotation marks.

TITLE= '*title-text*'
inserts the text string that you specify as the title into the metadata of a file.

title-text
is the text in the metadata of a file that indicates the title.

Requirement: You must enclose a *title-text* in quotation marks.

TOC_DATA | NOTOC_DATA
specifies whether contents data is embedded in the RTF file as hidden text.

NOTOC_DATA
instructs ODS not to insert contents data into the RTF file.

TOC_DATA
instructs ODS to insert contents data into the RTF file.

Tip: Insertion of table of contents data can be resumed in the middle of a SAS program by including the following statement:

```
ods rtf toc_data;
```

Default: NOTOC_DATA

Tip: To create a visible table of contents from the inserted table of contents data, specify the CONTENTS option.

See also: CONTENTS option

Featured in: Example 1 on page 256

TRANTAB= *translation-table*
specifies the translation table for ODS to use when it transcodes a file for output.

See: For information about the TRANTAB= option see *SAS National Language Support (NLS): Reference Guide*.

Details

Opening and Closing the RTF Destination You can modify an open RTF destination with many ODS RTF options. However, the FILE= option automatically closes the open destination that is referred to in the ODS RTF statement and closes any files that are associated with it, and then opens a new instance of the destination. If you use one of these options, you should explicitly close the destination yourself.

Understanding How RTF Formats Output RTF produces output for Microsoft Word 2002. Although other applications can read RTF files, the RTF output might not work successfully with the other applications.

The RTF destination enables you to view and edit the RTF output. ODS does not define the "vertical measurement," which means that SAS does not determine the optimal place to position each item on the page. For example, page breaks are not always fixed because you do not want your RTF output tables to split at inappropriate places when you edit your text. Your tables remain intact on one page, or break where you specify.

However, Microsoft Word needs to know the widths of table columns; and Microsoft Word cannot adjust tables if they are too wide for the page. Therefore, ODS measures the width of the text and tables (horizontal measurement). All of the column widths can be set properly by SAS and the table can be divided into panels if it is too wide to fit on a single page.

In short, when producing RTF output for input to Microsoft Word, SAS determines the horizontal measurement and Microsoft Word controls the vertical measurement. Because Microsoft Word can determine how much room there is on the page, your tables display consistently even after you modify your RTF file.

However, in SAS version 9.2, the ODS Measured tagset is introduced. This tagset enables users to specify how and where page breaks occur and when to place titles and footnotes into the body of a page. Refer to "ODS TAGSETS.RTF Statement" on page 286 for information on using Measured RTF.

Note: Complex tables that contain a large number of observations can reduce system efficiencies and take longer to process. △

ODS RTF and Graphics ODS RTF produces output in rich text format, which supports three formats for graphics that MS Word can read.

Format for Graphics	Corresponding SAS Graphics Driver
emfblips	SASEMF
pngblips	PNG
jpegblips	JPEG

When you do not specify a target device, the default target is SASEMF. You can also use the ACTIVEX, ACTXIMG, JAVAIMG graphics drivers to generate graphics in your

RTF documents. The ACTIVEX driver generates an ActiveX control. The ACTXIMG and JAVAIMG drivers generate PNG files with the ACTIVEX Control or JAVA Applets appropriately. For more information about graphics devices, see *SAS/GRAPH: Reference*.

Note: When you specify the JAVA device in the ODS RTF statement, the JAVAIMG driver is used. △

Examples

Example 1: Creating a Table of Contents from Embedded Data

ODS features:

ODS RTF statement:

Action:

CLOSE

Options:

CONTENTS

NOTOC_DATA

TOC_DATA

Other SAS features:

#BYVAL parameter in titles

NOBYLINE|BYLINE system option

OPTIONS statement

PROC FORMAT

PROC PRINT

PROC SORT

PROC REPORT

PROC TABULATE

TITLE statement

Data set:

See “Creating the Grain_Production Data Set” on page 878.

Format:

See “Creating the $CNTRY Format” on page 869.

Program Description The following example creates a table of contents page that contains embedded table of contents data for some procedures but not for others. The insertion of the table of contents data can be turned on and off in the middle of a program.

Program

Sort the data set Grain_Production. PROC SORT sorts the data, first by values of the variable Year, then by values of the variable Country, and finally by values of the variable Type.

```
proc sort data=Grain_Production;
   by year country type;
run;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create RTF output and create a new body file for each page of output. The ODS RTF statement opens the RTF destination and creates RTF output. The CONTENTS option creates a table of contents page that contains a Table of Contents field, which puts all of the contents information that is embedded in the document into a table of contents. However, the table of contents information is not embedded by default into your RTF file. The default is NOTOC_DATA. The embedded TOC data is not shown until you specify the option TOC_DATA.

```
ods rtf file='Grain.Rtf' contents toc_data;
```

Replace the default BY line with a new value in the BY line. The NOBYLINE option suppresses the default BY line variable. The #BYVAL parameter specification inserts the current value of the BY variable Year into the title.

```
options nobyline;
title 'Leading Grain-Producing Countries';
title2 'for #byval(year)';
```

Produce a report. This PROC REPORT step produces a report on grain production. Each BY group produces a page of output, and ODS creates a new body file for each BY group. The NOWINDOWS option instructs PROC REPORT to run without the REPORT window and to send its output to the open output destinations.

```
proc report data=Grain_Production nowindows;
   by year;
   column country type kilotons;
   define country  / group width=14 format=$cntry.;
   define type     / group 'Type of Grain';
   define kilotons / format=comma12.;
   footnote 'Measurements are in metric tons.';
run;
```

Restore the default BY line and clear the second TITLE statement. The BYLINE option restores the default BY line. The TITLE2 statement clears the second TITLE statement.

```
options byline;
title2;
```

Suppress the insertion of table of contents data into the RTF file. The NOTOC_DATA option instructs ODS not to insert the table of contents data into the RTF file. There will be no entry for the TABULATE procedure in the table of contents page.

```
ods rtf notoc_data;
```

Produce a report. The TABLE statement in the PROC TABULATE step uses three dimensions. Year defines pages, Country and Type define the rows, and Kilotons defines the columns. Therefore, PROC TABULATE explicitly produces one page of output for 1995 and one page for 1996, based on the years specified in the Grain_Production data set. ODS also starts a new body file for each page.

```
proc tabulate data=Grain_Production format=comma12.;
   class year country type;
```

```
   var kilotons;
   table year,
         country*type,
         kilotons*sum=' ' / box=_page_ misstext='No data';
   format country $cntry.;
   footnote 'Measurements are in metric tons.';
run;
```

Enable the insertion of table of contents data into the RTF file. The TOC_DATA option instructs ODS to insert the table of contents data into the RTF file. There will be an entry for the PRINT procedure in the table of contents page.

```
ods rtf toc_data;
```

Print the Grain_Production DATA set.

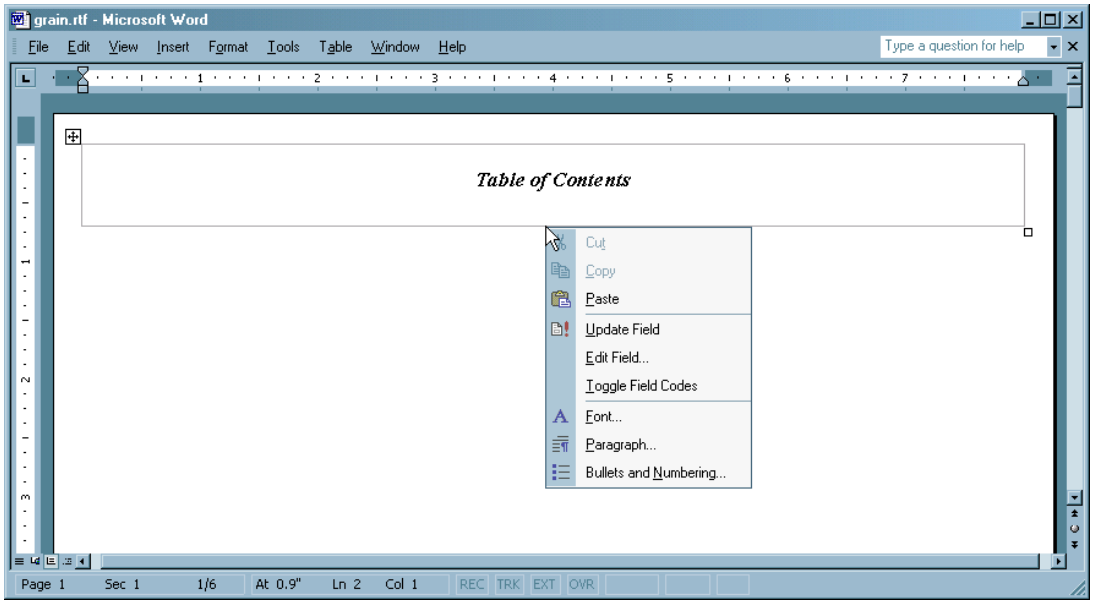
```
proc print data=Grain_Production;
run;
```

Close the RTF destination. The ODS RTF CLOSE statement closes the RTF destination and all the files that are associated with it. If you do not close the destination, you cannot view the files in a browser window.

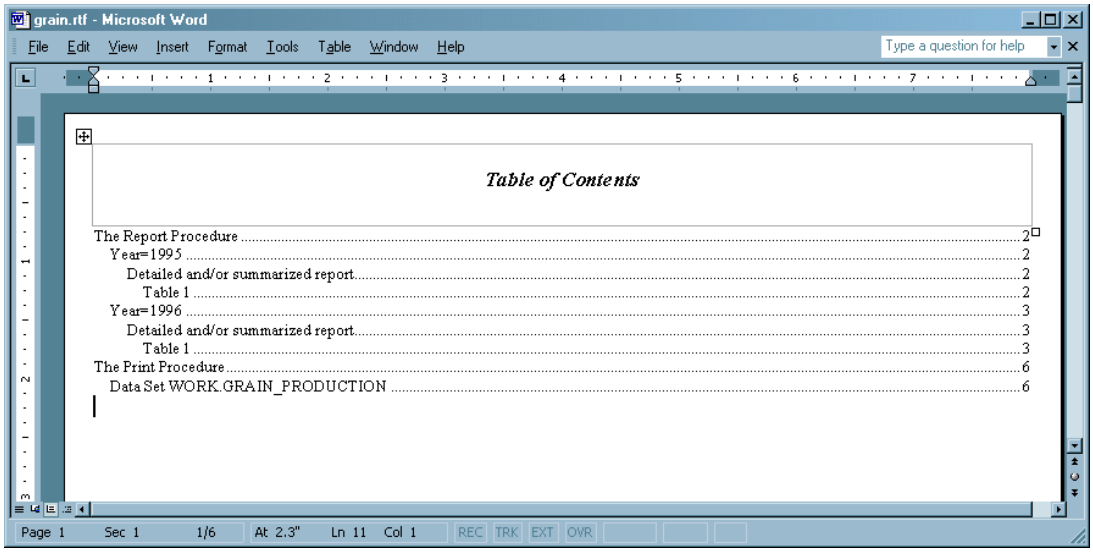
```
ods rtf close;
```

RTF Output

By default the table of contents is collapsed on the table of contents page. To expand the table of contents, right-click under the title in Microsoft Word and select "Update Field" from the selection list.



The table of contents contains only entries for PROC REPORT and PROC PRINT. By default the table of contents data is not embedded in the RTF document. To embed the table of contents data, specify the TOC_DATA option, which results in an entry for PROC REPORT. If you specify the NOTOC_DATA option before the TABULATE procedure, ODS does not insert contents information into the RTF document, and no entry for PROC TABULATE appears in the table of contents. If you specify the TOC_DATA option before the PRINT procedure, ODS inserts contents data, and an entry for PROC PRINT appears in the table of contents.



Example 2: Justifying Title and Footnotes When You Specify the BODYTITLE_AUX Option

ODS features:

ODS RTF statement:

Action:

CLOSE

Options:

BODYTITLE_AUX

FILE=

Other SAS features:

OPTIONS statement

PROC PRINT

TITLE statement

Program Description When you want to place the titles and footnotes in the body of the RTF output, you usually specify the BODYTITLE option. However, to center your titles and footnotes or to justify them, you need to specify the BODYTITLE_AUX option. The following example shows how to left justify, right justify, and center titles and footnotes in the body of the output.

Note: The preferred way to accomplish this functionality is to use the measured ODS TAGSETS.RTF statement that was introduced in 9.2. Refer to "ODS TAGSETS.RTF Statement" on page 286. △

Program

Specify the layout of the output. Instruct ODS not to print the date or time on the page and not to write any SAS statistics to the SAS log. Set the page size to 60 and the line size to 78.

```
OPTIONS NODATE NOSTIMER LS=78 PS=60;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create RTF output. The ODS RTF statement opens the RTF destination and creates RTF output. The BODYTITLE_AUX option tells SAS to place the titles and footnotes in the body of the output. Additionally, this option places the titles and footnotes into cells.

```
ods rtf file="bodytitle_aux.rtf" bodytitle_aux;
```

Print the SASHELP DATA set.

```
proc print data=sashelp.class;
run;
```

Add titles and footnotes to the output. Because you have specified the BODYTITLE_AUX option, ODS adds the titles and footnotes to the body of the output and places the text into cells. The J= style specifies the position of the title and footnote text on the page: left, center, or right.

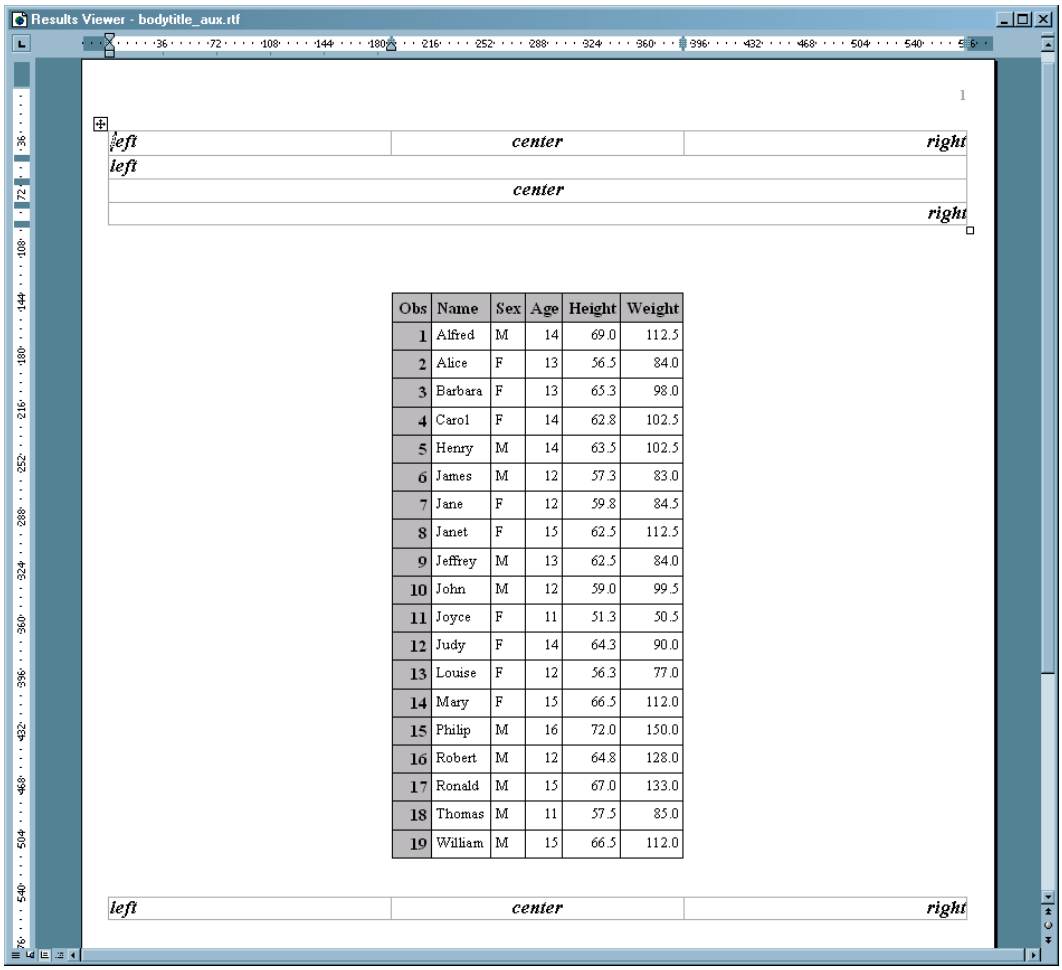
```
title j=l "left" j=c "center" j=r "right";
 title2 j=l "left";
 title3 j=c "center";
 title4 j=r "right";
 footnote j=l "left" j=c "center" j=r "right";
run;
```

Close the RTF destination. The ODS RTF CLOSE statement closes the RTF destination and all the files that are associated with it. If you do not close the destination, you cannot view the files in a browser window.

```
ods rtf close;
```

RTF Output

The following output shows how ODS places the titles and footnotes into the body of the output when you specify the BODYTITLE_AUX option. The text of the titles and footnotes are then placed into cells and tables. The JUSTIFY style element is then used to center, right justify, or left justify the title and footnote text.



Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Example 3: RTF Interaction with the ORIENTATION= System Option

ODS features:
- ODS RTF statement:
  - Action:
    - CLOSE
  - Options:
    - FILE=

Other SAS features:
- OPTIONS statement:
  - ORIENTATION option
- PROC PRINT
- TITLE statement

Program Description When you want to change the RTF page orientation, specify the ORIENTATION= system option. To activate or trigger this change of the page orientation, the ODS RTF statement needs to follow the ORIENTATION= option. The following example provides example code for specifying a page orientation change within an RTF file.

Program

Specify the layout of the output. Instruct ODS not to print the date or time on the page and not to write any SAS statistics to the SAS log. Set the page size to 60 and the line size to 78.

```
OPTIONS NODATE NOSTIMER LS=78 PS=60;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Add titles and footnotes to the output. Add a title for the overall file output and then titles that describe the changing orientation.

```
title 'Page Orientation';
title2 'Default';
```

Create RTF output. The ODS RTF statement opens the RTF destination and creates RTF output.In this case, the statement also triggers the change in the page orientation from the default.

```
ods rtf file="ChgOrientation.rtf";
```

Print the SASHELP DATA set with only one observation. The page orientation is the default orientation which is portrait.

```
proc print data=sashelp.class (obs=1);
run;
```

Add a title to change the page orientation in the output file. Add a title to change the page orientation to landscape.

```
title 'Page Orientation';
title2 'Landscape';
```

Specify the system option that will change the page orientation.

```
options orientation=landscape;
```

Trigger the page orienation change. This RTF statement triggers the change of the page orientation from protrait to landscape.

```
ods rtf;
```

Print the SASHELP DATA set with only one observation.

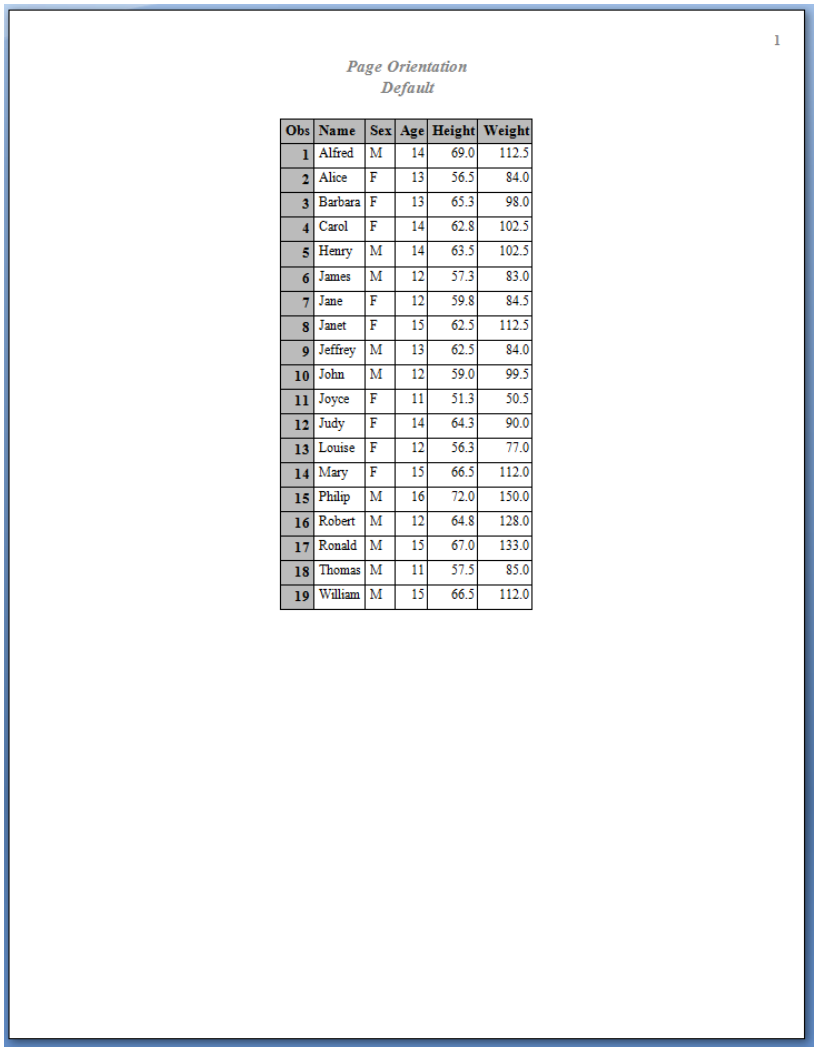
```
proc print data=sashelp.class (obs=1);
run;
```

Close the RTF destination. The ODS RTF CLOSE statement closes the RTF destination and all the files that are associated with it. If you do not close the destination, you cannot view the files in a browser window.

```
ods rtf close;
```

RTF Output

The following shows the RTF output for the first page. The orientation is portrait, which is the default.

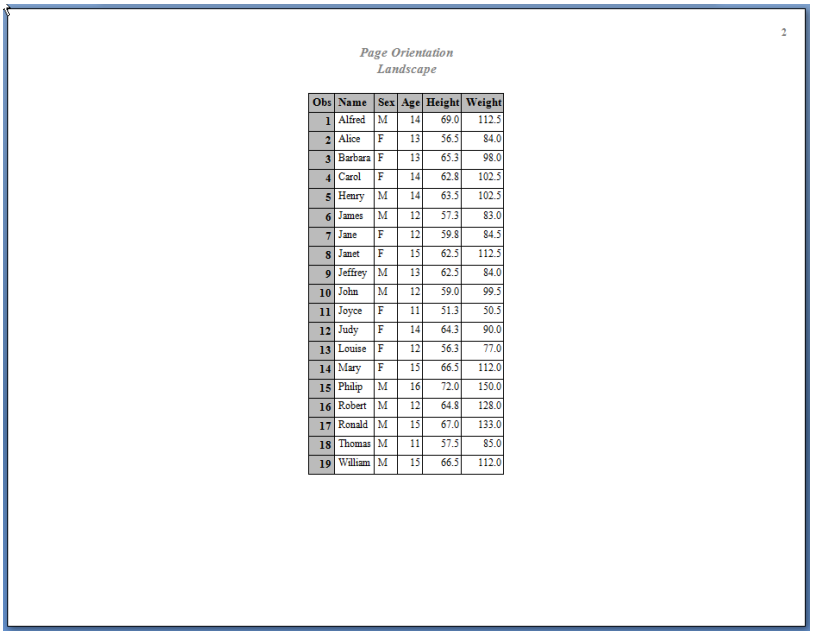


1

Page Orientation
Default

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

The following shows the RTF output for the second page. The orientation was changed to landscape.



2

Page Orientation
Landscape

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

ODS SELECT Statement

Specifies output objects for ODS destinations.

Valid: anywhere

Category: ODS: Output Control

See also: “ODS EXCLUDE Statement” on page 110

Tip: You can maintain a selection list for one destination and an exclusion list for another. However, it is easier to understand the results if you maintain the same types of lists for all of the destinations to which you route output.

Syntax

ODS <*ODS-destination*> **SELECT** *selection(s)* | ALL | NONE;

Required Arguments

selection(s)
specifies output objects to add to a selection list. ODS sends the items in the selection list to all active ODS destinations. By default, ODS automatically modifies selection lists when a DATA step that uses ODS or a procedure step ends. For information about modifying these lists, see “Selection and Exclusion Lists” on page 34. For information about ending DATA and procedure steps, see the section on DATA Step Processing in *SAS Language Reference: Concepts*.

Each *selection* has the following form:

output-object <(PERSIST)>

output-object
specifies the output object to select.

To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that your SAS program produces. You can specify an output object as one of the following:

- a full path. For example,

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  is the full path of the output object.
- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  then the partial paths are:

  ```
  City_Pop_90.TestsForLocation
  TestsForLocation
  ```
- a label that is enclosed by quotation marks.

  For example,

  ```
  "The UNIVARIATE Procedure"
  ```
- a label path. For example, the label path for the output object is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```

  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement. △
- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```

  then the partial label paths are:

  ```
  "CityPop_90"."Tests For Location"
  "Tests For Location"
  ```
- a mixture of labels and paths.
- any of the partial path specifications, followed by a pound sign (#) and a number. For example, **`TestsForLocation#3`** refers to the third output object that is named **`TestsForLocation`**.

Restriction:

See also: "ODS TRACE Statement" on page 317

(PERSIST)
keeps the *output-object* that precedes the PERSIST option in the selection list, even if the DATA or procedure step ends, until you explicitly modify the list with one of the following:

- any ODS EXCLUDE statement
- ODS SELECT NONE
- ODS SELECT ALL
- an ODS SELECT statement that applies to the same output object but does not specify PERSIST

Requirement: You must enclose PERSIST in parentheses.

ALL
specifies that ODS send all of the output objects to the open destination.

Alias: ODS SELECT DEFAULT

Interaction: If you specify ALL without specifying a destination, ODS sets the overall list to SELECT ALL and sets all other lists to their defaults.

NONE
specifies that ODS does not send any output objects to the open destination.

Interaction: If you specify NONE and you do not specify a destination, ODS sets the overall list to SELECT NONE and sets all other lists to their defaults.

Tip: Using the NONE action is different from closing a destination. The output destination is still open, but ODS restricts the output that it sends to the destination.

Tip: To temporarily suspend a destination, use ODS SELECT NONE. Use ODS SELECT ALL when you want to resume sending output to the suspended destination.

Options

NOWARN
suppresses the warning that an output object was requested but not created.

ODS-destination
specifies to which ODS destination's selection list to write, where *ODS-destination* can be any valid ODS destination except for the OUTPUT destination. For a discussion of ODS destinations, see "Understanding ODS Destinations" on page 24.

Default: If you omit *ODS-destination*, ODS writes to the overall selection list.

Restriction: You cannot write to the OUTPUT destination's selection list.

Tip: To set the selection list for the Output destination to something other than the default, see the "ODS OUTPUT Statement" on page 184.

WHERE=*where-expression*
selects output objects that meet a particular condition. For example, the following statement selects only output objects with the word "Histogram" in their name:

```
ods select where=(_name_ ?  'Histogram');
```

where-expression

is an arithmetic or logical expression that consists of a sequence of operators and operands. *where-expression* has this form:

(*subsetting-variable* <*comparison-operator where-expression-n*>)

subsetting-variable
Subsetting variables are a special kind of WHERE expression operand used by SAS to help you find common values in items. For example, this ODS SELECT statement selects only output objects with the path **City_Pop_90.TestsForLocation** :

```
ods select  / where=(_path_ = 'City_Pop_90.TestsForLocation' );
```

subsetting-variable is one of the following:

LABEL
is the label of the output object

LABELPATH
is the label path of the output object

NAME
is the name of the output object.

PATH
is the full or partial path of the output object.

operator
compares a variable with a value or with another variable. *operator* can be AND, OR NOT, OR, AND NOT, or a comparison operator.
The following table lists some comparison operators:

Table 5.31 Examples of Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Example

Example 1: Using a Selection List with Multiple Procedure Steps

ODS features:
ODS SELECT statement:
with label
with name
with and without PERSIST
ALL

ODS SHOW statement
ODS HTML statement:
BODY=
CONTENTS=
FRAME=
PAGE=

Other SAS features:
PROC GLM
PROC PRINT
PROC PLOT

Data Sets:
See “Creating the Iron Data Set” on page 879.

This example runs the same procedures multiple times to illustrate how ODS maintains and modifies a selection list. The ODS SHOW statement writes the overall selection list to the SAS log. The example does not alter selection lists for individual destinations. The contents file that is generated by the ODS HTML statement shows which output objects are routed to both the HTML and the LISTING destinations.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 903. △

This example creates and prints data sets from the parameter estimates that PROC GLM generates. This procedure is part of SAS/STAT software.

Program

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from the procedures is sent to the file **odspersist-body.htm**. The FRAME=, CONTENTS=, and PAGE= options create the files OdsPersist-Frame.htm, OdsPersist-Contents.htm, and OdsPersist-Page.htm, respectively. These files, together with the file OdsPersist-Body.htm, create a frame that includes a table of contents and a table of pages that link to the contents of the body file.

```
ods html body='odspersist-body.htm'
        frame='odspersist-frame.htm'
     contents='odspersist-contents.htm'
         page='odspersist-page.htm';
```

Write the overall selection list to the SAS log. The ODS SHOW statement writes to the SAS log the overall list, which is set to SELECT ALL by default. See [1] in “SAS Log” on page 271.

```
ods show;
```

Specify the output objects that will be sent to the open destinations. The ODS SELECT statement determines which output objects ODS sends to the LISTING and HTML destinations. In this case, ODS sends all output objects that are named ParameterEstimates and all output objects that are labeled “Type III Model ANOVA” to the two destinations.

```
ods select ParameterEstimates
           "Type III Model ANOVA";
```

Write the modified overall selection list to the SAS log. The ODS SHOW statement writes to the SAS log the overall selection list, which now contains the two items that were specified in the ODS SELECT statement. See [2] in the "SAS Log" on page 271.

```
ods show;
```

Create the output objects and send the selected output objects to the open destinations. As PROC GLM sends each output object to the Output Delivery System, ODS sends the two output objects from PROC GLM that match the items in the selection list to the open destinations. See **1.** in the table of contents in "HTML Output" on page 273. Note that it is the label of an output object, not its name, that appears in the table of contents. The label for ParameterEstimates is "Solution".

```
proc glm data=iron;
   model loss=fe;
   title 'Parameter Estimates and Type III Model ANOVA';
run;
```

Write the overall selection list to the SAS log. PROC GLM supports run-group processing. Therefore, the RUN statement does not end the procedure, and ODS does not automatically modify the selection list. See [3] in the "SAS Log" on page 271.

```
ods show;
```

End the GLM procedure. The QUIT statement ends the procedure. ODS removes all objects that are not specified with PERSIST from the selection list. Because this action removes all objects from the list, ODS sets the list to its default, SELECT ALL.

```
quit;
```

Write the current selection list to the SAS log. The ODS SHOW statement writes the current selection list to the SAS log. See [4] in the "SAS Log" on page 271.

```
ods show;
```

Create the output objects, send the selected output objects to the open destinations, and end the procedure. As PROC GLM sends each output object to the Output Delivery System, ODS sends all the output objects to the HTML and LISTING destinations. See **2.** in the table of contents in "HTML Output" on page 273.

The QUIT statement ends the procedure. Because the list uses the argument ALL, ODS does not automatically modify it when the PROC step ends.

```
proc glm data=iron;
   model loss=fe;
   title 'All Output Objects Selected';
run;
quit;
```

Modify the overall selection lists. This ODS SELECT statement modifies the overall selection list so that it sends all output objects that are named OverallANOVA, and all output objects that are labeled Fit Statistics, to both the HTML and LISTING destinations. The PERSIST option specifies that OverallANOVA should remain in the selection list when ODS automatically modifies it.

```
ods select OverallANOVA(persist) "Fit Statistics";
```

Create the output objects and send the selected output objects to the open destinations. As PROC GLM sends each output object to the Output Delivery System, ODS sends the two output objects from PROC GLM that match the items in the selection list to the HTML and LISTING destinations. See **3.** in the table of contents in "HTML Output" on page 273.

```
proc glm data=iron;
   model loss=fe;
   title 'OverallANOVA and Fitness Statistics';
run;
```

End the GLM procedure and automatically modify the selection list. When the QUIT statement ends the procedure, ODS automatically modifies the selection list. Because OverallANOVA was specified with the PERSIST option, it remains in the selection list. Because Fitness Statistics was not specified with the PERSIST option, ODS removes it from the selection list.

```
quit;
```

Write the current selection list to the SAS log. The ODS SHOW statement writes the current selection list to the SAS log. See [5] in the "SAS Log" on page 271.

```
ods show;
```

Create the output objects and send the selected output objects to the open destinations. As PROC GLM sends each output object to the Output Delivery System, ODS sends only the output object that is named OverallANOVA to the HTML and LISTING destinations. See **4.** in the table of contents in "HTML Output" on page 273.

```
proc glm data=iron;
   model loss=fe;
   title 'OverallANOVA';
   title2 'Part of the Selection List Persists';
run;
```

End the GLM procedure and automatically modify the selection list. When the QUIT statement ends the procedure, ODS automatically modifies the selection list. Because OverallANOVA was specified with the PERSIST option, it remains in the selection list.

```
quit;
```

PROC PRINT does not produce any output that is named OverallANOVA. Therefore, no PROC PRINT output is sent to the ODS destinations.

```
proc print data=iron;
   title 'The IRON Data Set';
run;
```

Reset all selection lists. This ODS SELECT statement resets all selection lists to their defaults.

```
ods select all;
```

Create the plots. As PROC PLOT creates and sends each output object to the Output Delivery System, ODS sends each one to the HTML and LISTING destinations because their lists and the overall list are set to SELECT ALL (the default).

```
proc plot data=iron;
   plot fe*loss='*' / vpos=25 ;
   label fe='Iron Content'
       loss='Weight Loss';
   title 'Plot of Iron Versus Loss';
run;
```

End the PLOT procedure. The QUIT statement ends the PLOT procedure. Because the list uses the argument ALL, ODS does not automatically modify the list when the PROC step ends.

```
quit;
```

Close the HTML destination. This ODS HTML statement closes the HTML destination and all the files that are associated with it.

```
ods html close;
```

SAS Log

Output 5.5 The ODS SHOW Statement Writes the Current Selection List to the SAS Log.

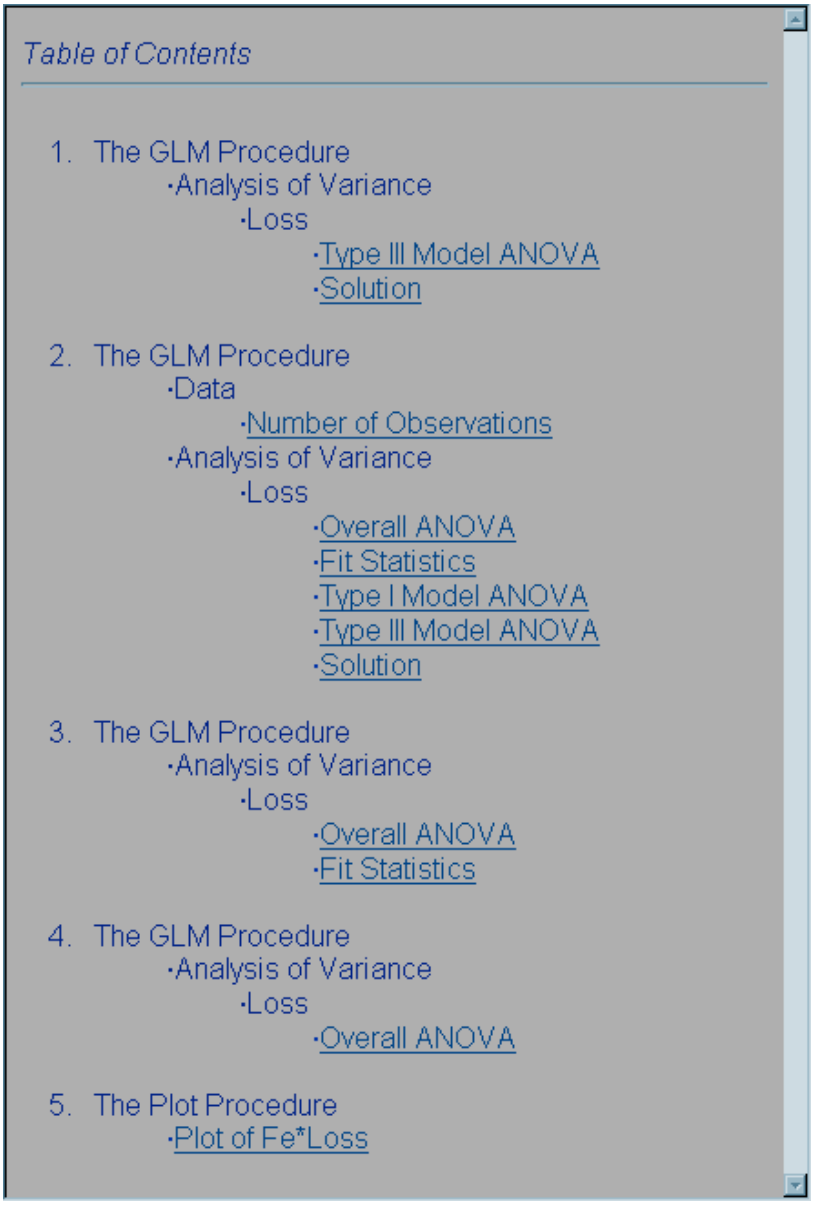
```
10   ods html body='odspersist-body.htm'
11        contents='odspersist-contents.htm'
12           frame='odspersist-frame.htm'
13            page='odspersist-page.htm';
NOTE: Writing HTML Body file: odspersist-body.htm
NOTE: Writing HTML Contents file: odspersist-contents.htm
NOTE: Writing HTML Pages file: odspersist-page.htm
NOTE: Writing HTML Frames file: odspersist-frame.htm
14   ods show;
Current OVERALL select list is: ALL   [1]
15   ods select ParameterEstimates
16              "Type III Model ANOVA";
17   ods show;
Current OVERALL select list is:    [2]
1. ParameterEstimates
2. "Type III Model ANOVA"
18   proc glm data=iron;
19      model loss=fe;
20      title 'Parameter Estimates and Type III Model ANOVA';
21   run;
22   ods show;
Current OVERALL select list is:    [3]
1. ParameterEstimates
2. "Type III Model ANOVA"
23   quit;
NOTE: PROCEDURE GLM used:
      real time           x.xx seconds
      cpu time            x.xx seconds

24   ods show;
Current OVERALL select list is: ALL   [4]
25   proc glm data=iron;
26      model loss=fe;
27      title 'All Output Objects Selected';
28   run;
29   quit;
NOTE: PROCEDURE GLM used:
      real time           x.xx seconds
      cpu time            x.xx seconds
```

```
30   ods select OverallANOVA(persist) "Fit Statistics";
31   proc glm data=iron;
32      model loss=fe;
33      title 'OverallANOVA and Fitness Statistics';
34   run;
35   quit;
NOTE: PROCEDURE GLM used:
      real time           x.xx seconds
      cpu time            x.xx seconds

36
37  ods show;
Current OVERALL select list is:    [5]
1. OverallANOVA(PERSIST)
38  proc glm data=iron;
39     model loss=fe;
40     title 'OverallANOVA';
41     title2 'Part of the Selection List Persists';
42  run;
43  quit;
NOTE: PROCEDURE GLM used:
      real time           x.xx seconds
      cpu time            x.xx seconds

44  proc print data=iron;
45     title 'The IRON Data Set';
46  run;
NOTE: PROCEDURE PRINT used:
      real time           x.xx seconds
      cpu time            x.xx seconds

47  ods select all;
48  proc plot data=iron;
49     plot fe*loss='*' / vpos=25 ;
50     label fe='Iron Content'
51         loss='Weight Loss';
52     title 'Plot of Iron Versus Loss';
53  run;
54  quit;
```

HTML Output

Display 5.23 Contents File Produced by the ODS HTML Statement

The contents file shows the output objects from each procedure that ODS sent to the open ODS destinations. You can see that no output was written to the HTML destination for PROC PRINT (because PROC PRINT did not produce anything whose name matched the name in the selection list). You can also see that the PROC PLOT output was written to the HTML destination after the ODS SELECT ALL statement was executed.



Example 2: Conditionally Selecting Output Objects

ODS features:

ODS SELECT statement:

WHERE= option

ODS TRACE statement:

LABEL option
EXCLUDED
ODS HTML statement

Other SAS features:
PROC UNIVARIATE

Program

Create the BPressure data set.

```
data BPressure;
   length PatientID $2;
   input PatientID $ Systolic Diastolic @@;
   datalines;
CK 120 50  SS 96  60 FR 100 70
CP 120 75  BL 140 90 ES 120 70
CP 165 110 JI 110 40 MC 119 66
FC 125 76  RW 133 60 KD 108 54
DS 110 50  JW 130 80 BH 120 65
JW 134 80  SB 118 76 NS 122 78
GS 122 70  AB 122 78 EC 112 62
HH 122 82
;
run;
```

Create HTML output and add a title.

```
ods html file='MyOutputObjects.html';
      title 'Systolic and Diastolic Blood Pressure';
```

Specify that SAS write the trace record to the SAS log. This ODS TRACE statement writes the trace record to the SAS log. The LABEL option includes label paths in the trace record. The EXCLUDED option includes information on output objects that SAS excludes from the output destination.

```
ods trace on / label excluded;
```

Select output objects. The ODS SELECT statement with the WHERE = option specified selects output objects that are named 'Moments' and that have 'Diastolic' in the path name.

```
ods select where=(_path_ ? "Diastolic" and  _name_='Moments') ;
```

Create the output objects and send the selected output objects to the open destinations. As PROC UNIVARIATE sends each output object to the Output Delivery System, ODS sends the output object from PROC UNIVARIATE that matches the items in the selection list to the open destinations.

```
proc univariate data=BPressure;
   var Systolic Diastolic;
run;
```

Close the HTML destination. This ODS HTML statement closes the HTML destination and all the files that are associated with it.

```
ods html close;
```

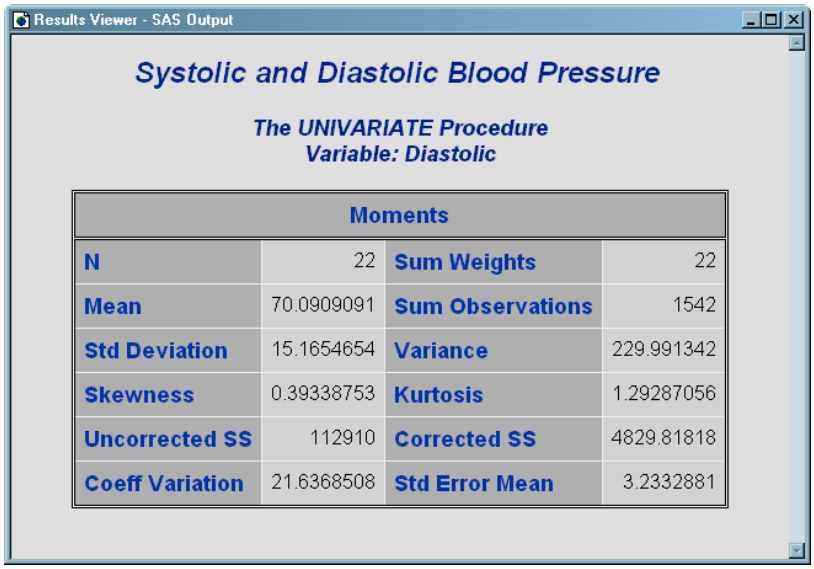
SAS Log: Trace Record

Output 5.6 Partial SAS Log Including Trace Record

```
Output Excluded:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.Systolic.Moments
Label Path: 'The Univariate Procedure'.'Systolic'.'Moments'
-------------

Output Excluded:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.Systolic.BasicMeasures
Label Path: 'The Univariate Procedure'.'Systolic'.
          'Basic Measures of Location and Variability'
-------------

Output Excluded:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.Systolic.TestsForLocation
Label Path: 'The Univariate Procedure'.'Systolic'.
          'Tests For Location'
-------------

Output Excluded:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.Systolic.Quantiles
Label Path: 'The Univariate Procedure'.
          'Systolic'.'Quantiles'
-------------

Output Excluded:
-------------
Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.Systolic.ExtremeObs
Label Path: 'The Univariate Procedure'.
          'Systolic'.'Extreme Observations'
-------------
```

```
Output Added:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.Diastolic.Moments
Label Path: 'The Univariate Procedure'.'Diastolic'.'Moments'
-------------

Output Excluded:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.Diastolic.BasicMeasures
Label Path: 'The Univariate Procedure'.'Diastolic
       '.'Basic Measures of Location and Variability'
-------------

Output Excluded:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.Diastolic.TestsForLocation
Label Path: 'The Univariate Procedure'.'Diastolic'.
          'Tests For Location'
-------------

Output Excluded:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.Diastolic.Quantiles
Label Path: 'The Univariate Procedure'.'Diastolic'.'Quantiles'
-------------

Output Excluded:
-------------
Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.Diastolic.ExtremeObs
Label Path: 'The Univariate Procedure'.'Diastolic'.
'Extreme Observations'
-------------
```

HTML Output



See Also

Statements:

"ODS EXCLUDE Statement" on page 110
"ODS SHOW Statement" on page 277
"ODS TRACE Statement" on page 317

ODS SHOW Statement

Writes the specified selection or exclusion list to the SAS log.

Valid: anywhere

Category: ODS: Output Control

Syntax

ODS <*ODS-destination*> **SHOW**;

Options

ODS-destination
specifies which ODS destination's selection or exclusion list to write to the SAS log. *ODS-destination* must be a valid ODS destination. For information about ODS destinations, see "Understanding ODS Destinations" on page 24. For information about selection and exclusion lists, see "Selection and Exclusion Lists" on page 34.

Default: If you omit *ODS-destination*, ODS SHOW writes the overall selection or exclusion list.

See Also

Statements:

"ODS EXCLUDE Statement" on page 110

"ODS SELECT Statement" on page 264

"ODS TRACE Statement" on page 317

ODS Tagset Statement

Opens, manages, or closes the specified tagset destination.

Valid: anywhere

Category: ODS: Third-Party Formatted

Interaction: If you use the ODS tagset statement in an ODS markup family statement that refers to an open ODS markup destination, then the option will force ODS to close the destination and all files associated with it, and then to open a new instance of the destination. For more information, see "Opening and Closing the MARKUP Destination" on page 167.

See also: For additional information about specifying tagsets, see Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795 or "ODS MARKUP Statement" on page 147.

Syntax

ODS *directory.tagset-name file-specification <option(s)>*;

ODS *directory.tagset-name file-specification action*;

Actions

The following table lists the actions available for the ODS tagset statement. For complete descriptions of actions see "Actions" on page 147 in the ODS MARKUP statement.

Table 5.32 ODS Tagset Action Summary Table

Task	Action
Close the destination and the file that is associated with it	CLOSE
Exclude output objects from the destination	EXCLUDE

Task	Action
Select output objects for the destination	SELECT
Write to the SAS log the current selection or exclusion list for the destination	SHOW

Options

The following table lists the options that are available for the ODS Tagset statement, which is part of the markup family of statements. For complete descriptions of these options, see "Options" on page 148 in the ODS MARKUP statement.

Table 5.33 ODS Tagset Option Summary Table

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view ODS output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=
Specify the character set to be generated in the META declaration for the output	CHARSET=
Open the tagset destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the tagset destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE

Task	Option
Specify markup tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify markup code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the markup files that the destination writes to	METATEXT=
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Open the destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an external file or a SAS catalog for all HTML files	PATH=
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

The following table lists the options that are most useful for the ODS tagset statement. You can use the ODS tagset options and the ODS MARKUP statement options together.

Table 5.34 ODS Tagset Option Summary Table

Task	Option
Specify tagset-specific options	OPTIONS
Specify a directory where the tagset is stored	*directory*
Specify a tagset	*tagset-name*

OPTIONS (DOC=) | *sub-option(s)*

specifies ODS tagset-specific sub-options and a named value.

(DOC='QUICK' | 'HELP' | 'SETTINGS')
provides information about the specified tagset.

QUICK
describes the options available for this tagset.

HELP
provides generic help and information with a quick reference.

SETTINGS
provides the current option settings.

Requirement: All values must be enclosed in quotation marks.

sub-option(s)
specifies one or more suboptions that are valid for the specified tagset. To list suboptions that are valid for a tagset, specify DOC="HELP" or DOC="QUICK" with the OPTIONS option.

Requirement: The OPTION suboptions must be enclosed in parenthesis.

Featured in: Example 1 on page 285

directory
specifies the directory where the specified tagset is stored. *directory* can be a directory supplied by SAS, a user- defined entry, or a libref. By default, the tagsets that SAS supplies are located in the directory TAGSETS, which is within the item store SASHELP.TMPLMST.

tagset-name
specifies the name of the tagset. *tagset-name* can be one of the following:

CHTML
produces compact, minimal HTML output that does not use style information. It does produce a hierarchical table of contents.

See: "ODS CHTML Statement" on page 85

CORE
contains a table of Unicode values and mnemonics. Refer to "Using Unicode Symbols" on page 99 for a detailed description on using this tagset.

CSV
produces tabular output that contains columns of data values that are separated by commas.

Interaction: The TEXT= option has no affect in the CSV file output.

Featured in: "Defining a Tagset Using SAS DATA Step Functions" on page 846.

CSVALL
produces HTML output containing columns of data values that are separated by commas, and produces tabular output with titles, notes, and bylines.

Interaction: The TEXT= option has no affect in the CSV file output.

See also: "ODS CSVALL Statement" on page 88

Feature in: Example 3 on page 173

CSVBYLINE
produces output with comma-separated values and columns of data that are separated by commas.

Interaction: The TEXT= option has no affect in the CSV file output.

DEFAULT

produces XML output.

DOCBOOK
produces XML output that conforms to the DocBook DTD by OASIS.

See also: “ODS DOCBOOK Statement” on page 91

ExcelXP
produces Microsoft spreadsheetML XML. This tagset is used to import data into Excel. Execute the following code to get detailed information on this tagset:

```
ods tagsets.excelxp file='test.xml' options(doc='help');
```

HTML4
produces HTML 4.0 embedded style sheets.

See also: “ODS HTML Statement” on page 124

HTMLCSS
produces HTML output with cascading style sheets that is similar to ODS HTML output.

See also: “ODS HTMLCSS Statement” on page 135

HTMLPANEL
creates panels for BY grouped graphs. It also has controls for semi-automatic and manually controlled paneling. This tagset makes it easy to put graphs and tables side-by-side on a page. Also included are controls for titles, footnotes, and bylines.

To get detailed help on this tagset, any of the following three lines of code can be executed:

```
ods tagsets.htmlpanel file=''gbypanel.html'' options(doc='help');


ods tagsets.htmlpanel  options(doc='quick');


ods tagsets.htmlpanel  options(doc='settings');
```

IMODE
produces HTML output as a column of output that is separated by lines. This tagset is used by the Japanese telephone service provider NTT.

See also: “ODS IMODE Statement” on page 140

MSOFFICE2K
produces HTML code for output generated by ODS for Microsoft Office products.

MVSHTML
produces URLs within HTML files that are used in the z/OS operating environment.

PHTML
produces simple HTML output that uses twelve style elements and no class attributes for the presentation. Class attributes are used only for the justification.

See also: “ODS PHTML Statement” on page 215

PYX
produces PYX, which is a simple, line-oriented notation used by Pyxie to describe the information communicated by an XML parser to an XML application. Pyxie is an Open Source library for processing XML with the Python programming language.

RTF

produces measured RTF. This tagset allows the user to specify how and where page breaks occur and when to place titles and footnotes into the body of a page. The RTF tagset enables SAS to place titles and footnotes into the body of the document so that it is outside of the control of Microsoft Word. Therefore, SAS becomes responsible for the implicit page breaks.

Refer to "ODS TAGSETS.RTF Statement" on page 286 for details on how to use the RTF tagset.

SASREPORT
causes embedded data to be produced in CSV format. SASREPORT11 and SASREPORT12 are the supported tagsets. For more information on how to use this tagset, execute one line of the following code:

```
ods tagsets.sasreport11 file='test.xml' options(doc='help');


ods tagsets.sasreport12 file='test.xml' options(doc='help');
```

user-defined-tagset
specifies the tagset that you created using PROC TEMPLATE.

Main discussion: "Creating Custom Tagsets" on page 842.

WML
uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a list of URLs as a table of contents.

See also: "ODS WML Statement" on page 326

WMLOLIST
uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with an option list for the table of contents. For more information, see Wireless Application Protocol.

XHTML
produces output in HTML format. For details on using this tagset, execute the following code:

```
ods tagsets.xhtml file='test.html' options(doc='help');
```

Note: There are also preproduction tagsets. These tagsets can be found at **http://support.sas.com** and are not supported by SAS. △

The following are diagnostic tagsets:

EVENT_MAP
creates XML output that shows which events are being triggered and which variables are used by an event to send output from a SAS process to an output file. When you run a SAS process with EVENT_MAP, ODS writes XML to an output file that shows all event names and variable names as tags. The output helps you to create your own tagsets.

NAMEDHTML
creates HTML output similar to STYLE_POPUP, but with all the objects labeled as they are when using ODS TRACE.

SHORT_MAP
creates a subset of the XML output that is created by the EVENT_MAP tagset.

STYLE_DISPLAY
creates a sample page of HTML output that is similar to STYLE_POPUP output. The output helps you to create and modify styles.

See also: STYLE_POPUP

STYLE_POPUP
creates HTML like HTMLCSS, but if you're using Internet Explorer, STYLE_POPUP displays a window that shows the resolved ODS style definition for any item that you select.

TEXT_MAP
creates text output that shows which events are being triggered as ODS handles the output objects.

Tip: You can use the TEXT_MAP output as an alternative to the output that is created by the EVENT_MAP tagset.

See also: EVENT_MAP

TPL_STYLE_LIST
creates HTML output in a bulleted list similar to EVENT_MAP but lists only a subset of the possible attributes.

Tip: The output helps you to understand tagsets and styles.

TPL_STYLE_MAP
creates XML output similar to EVENT_MAP but lists only a subset of the possible attributes.

Tip: The output helps you to understand tagsets and styles.

Details

Understanding Tagsets A tagset is a type of template that defines how to generate a markup language output type from SAS data. A markup language is a set of tags and format codes that are embedded in text in order to define layout and certain content.

Starting with SAS 8.2 software, you can use the ODS tagset statement to specify a tagset to create markup language output from the Output Delivery System. SAS provides tagset definitions for a variety of markup language output. For example, there are several SAS tagsets for XML output, HTML output, XSL, and so on. In addition to using the tagsets provided by SAS, you can modify the SAS tagsets, and you can create your own. By supplying new tagset definitions, ODS output and XML LIBNAME engine output is user-configurable, generating a wider variety of markup language output. For information on modifying SAS tagsets and creating your own tagsets, see Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795.

Listing Tagset Names To see a list of available tagsets, issue the following SAS statements or view them in the Templates window.

□ *Templates window:*

To display a list of the available tagsets using the SAS Explorer window, follow these steps:

1. From any window in an interactive SAS session, select **View ▸ Results**.
2. In the Results window, select **View ▸ Templates**.
3. In the Templates window, select and open **`Sashelp.tmplmst`**.
4. Select and open the **`Tagsets`** folder, which contains a list of available tagsets. If you want to view the underlying SAS code for a tagset, then select the tagset and open it.

Operating Environment Information: For information on navigating in the Explorer window without a mouse, see "Window Controls and General Navigation" in the SAS documentation for your operating environment. △

- *TEMPLATE procedure:*

  You can also display a list of the available tagsets by submitting the following PROC TEMPLATE statements:

```
proc template;
  list tagsets;
quit;
```

By default, PROC TEMPLATE lists the tagsets in SASHELP.TMPLMST and SASUSER.TEMPLAT. Typically, SASHELP.TMPLMST is a read-only item store for the SAS tagsets, and SASUSER.TEMPLAT is the item store for user-defined tagsets.

Viewing the Source of a Tagset To see the source for a tagset definition, you can either open the tagset in the SAS Explorer window, or use PROC TEMPLATE and specify the two-level name of the tagset. For example, to see the source of the SAS tagset CHTML, issue these SAS statements:

```
proc template;
  source tagsets.chtml;
quit;
```

Viewing Available Options for a Tagset To view the options that are available for a specific tagset, use the OPTIONS (DOC=) option with one of the following specified:

QUICK
: describes the options available for this tagset.

HELP
: provides generic help and information with a quick reference.

SETTINGS
: provides the current option settings.

Examples

Example 1: Using the DOC Suboption to Get ODS TAGSETS.HTMLPANEL Information

ODS features:
    ODS TAGSETS.HTMLPANEL statement:
        Action:
            CLOSE
        Options:
            OPTIONS
                (DOC="HELP")
            FILE=
Other SAS features:
    PROC PRINT

Program Description The following example prints to the SAS log the OPTIONS suboptions for the HTMLPANEL tagset and a description of each available suboption.

Program

Print information about the OPTIONS suboptions to the SAS log file. Specifying the OPTIONS suboption (DOC='HELP') prints Help for the ODS TAGSETS.HTMLPANEL statement suboptions to the SAS log file. The FILE= option prints the data results to an RTF file named Help.rtf.

```
ods tagsets.htmlpanel file='Help.rtf' options (doc="help");
```

Print the data set SASHELP.CLASS. The PROC PRINT statement prints the SASHELP.CLASS data set.

```
proc print data=Sashelp.Class;
run;
```

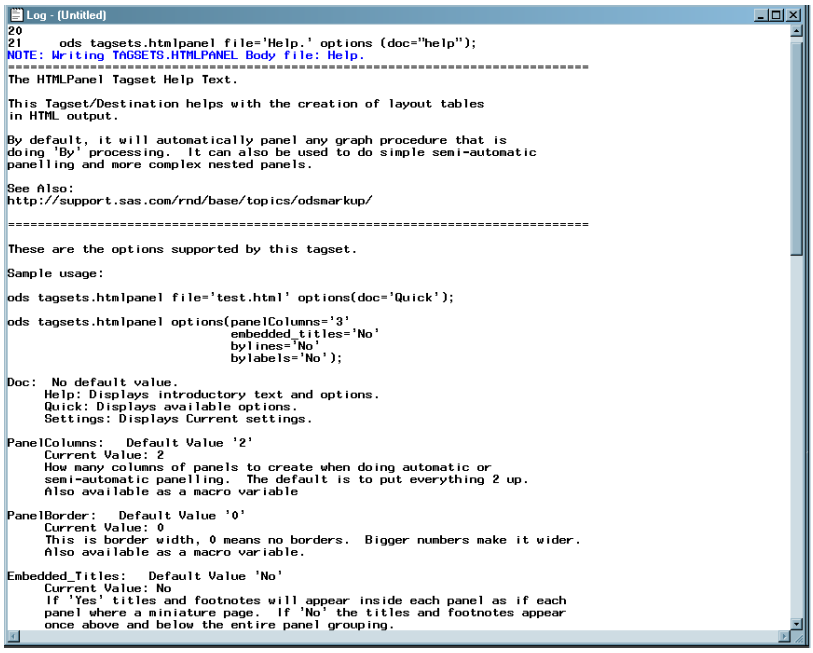
Close all destinations. Close the ODS TAGSETS.HTMLPANEL destination and any other open destinations. This statement also closes all the files that are associated with each open destination. If you do not close a destination, then you cannot view the files in a browser window.

```
ods _all_ close;
```

Output

Display 5.24 Options Available for the HTMLPANEL Tagset

Specify the "DOC='HELP' suboption to print all of the OPTIONS suboptions and information about each of the suboptions to the SAS log.



See Also

Statement:

"ODS MARKUP Statement" on page 147

Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795

ODS TAGSETS.RTF Statement

Opens, manages, or closes the RTF destination, which produces measured output that is written in Rich Text Format for use with Microsoft Word 2002.

Valid in: anywhere

Category: ODS: Third-Party Formatted

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment that does not have SAS software installed, this output will not be displayed correctly. The SAS Monospace font is not recognized if SAS is not installed. For the correct display of your document, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
```

Tip: Microsoft Word 2002 is the current official minimum level that is supported. However, no problems have been found with Microsoft Word 2000 and SAS RTF files.

Syntax

ODS TAGSETS.RTF <(<ID=> *identifier*)> *action*;

ODS TAGSETS.RTF <(<ID=> *identifier*)> <*option(s)*>;

Actions

The following table lists the actions that are available for the ODS RTF statement and for the ODS TAGSETS.RTF statement. For a complete description, see “Actions” on page 243 in the ODS RTF Statement.

Table 5.35 ODS TAGSETS.RTF Action Summary Table

Task	Action
Close the RTF destination and the file that is associated with it	CLOSE
Exclude output objects from the RTF destination	EXCLUDE
Select output objects for the RTF destination	SELECT
Write to the SAS log the current selection or exclusion list for the RTF destination	SHOW

Options

The following table lists a subset of the options that traditional ODS RTF statement also supports. For a complete description of these options, see “Options” on page 243 in the ODS RTF Statement.

Note: The BODYTITLE and SAS DATE options are supported options for the traditional ODS RTF statement. However, they are not supported options for the ODS TAGSETS.RTF statement because their functionality is built into the ODS TAGSETS.RTF statement. △

Table 5.36 Options Supported for ODS RTF and ODS TAGSETS.RTF

Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify the text string that identifies the author. This identifier is inserted into the metadata of a file.	AUTHOR=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Specify the number of columns to create on each page of output	COLUMNS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Specify a device for the RTF output destination	DEVICE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Open the ODS RTF destination and specify the name of the file to which to write information	FILE=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Open multiple instances of the same destination at the same time	ID=
Specify the image resolution for the graphical output	IMAGE_DPI=
Control where tables split on a page	KEEPN | NOKEEPN
Create a new body file at the specified starting point	NEWFILE=
Suppress currently defined footnotes in the graphics file. They appear in the RTF file instead.	NOGFOOTNOTE
Suppress currently defined titles in the graphics file. They appear in the RTF file instead.	NOGTITLE
Insert the text that you specify into the metadata of the RTF file	OPERATOR=
Specify that the output from the destination be added to an ODS package	PACKAGE
Specify the location of an aggregate storage location or a SAS catalog for all RTF files	PATH=
Specify an alternative character or string to separate lines in the output file	RECORD_SEPARATOR=
Control page breaks	STARTPAGE=
Specify a style definition to use when writing the RTF files	STYLE=
Insert text into your RTF output	TEXT=

Task	Option
Insert the text string that you want as your title into the metadata of a file	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

The following table lists the options available exclusively for the ODS TAGSETS.RTF statement. You can use the ODS RTF options and the ODS TAGSETS.RTF options together.

Table 5.37 ODS TAGSETS.RTF Option Summary Table

Task	Option
Specify TAGSETS.RTF specific options	OPTIONS
Specify the number of panels that will be rendered for a multipanel table	PAGEPANELS=
Specify the number of rows that will be rendered in a table	TABLEROWS=
Specify that every page of a table is formatted the same	UNIFORM

OPTIONS (CONTENTS= | DOC= | SECT= | TABLES_OFF= |TOC_DATA= |TROWD= |TRHDR= | TROWHDRCELL=)

specifies ODS TAGSETS.RTF specific suboptions and a named value.

(CONTENTS= 'YES')

produces a table of contents (TOC) page for RTF documents that are opened in Microsoft Word. The table of contents page contains a Table of Contents field that puts all of the contents information that is embedded in the document into a table of contents. To display the captured TOC data, you must turn on the TOC_DATA option. To expand the table of contents right-click under the title in Microsoft Word and select **`Update Field`** from the selection list.

Note: From Microsoft Word, you might need to right-click lower on the page to get the **`Update Field`** value to appear in the selection list. △

YES — adds a table of contents page to the top of the RTF file. This table of contents page is followed by a page break.

Alias: ON

Requirement: All values must be enclosed in quotation marks.

Tip: To go to a specific topic in the document you can double-click or hold down the CTRL key and click on the topic in the table of contents. You might have to configure Microsoft Word to use the CTRL-click method. Select **Tools ▶ Options ▶ Edit** and then check **`Use CTRL + Click to follow hyperlink`**.

Tip: The TOC_DATA option must be set to YES to capture TOC data. If you specify CONTENTS=YES, but you do not specify TOC_DATA, no Table of Contents data is captured. The error displayed on the Table of Contents page is "Error! No table of contents entries found".

See: Suboption TOC_DATA for details on displaying the contents embedded in the document.

Featured in: Example 1 on page 295

(DOC='QUICK' | 'HELP' | 'SETTINGS')
provides information about the tagset.

QUICK
describes the options available for this tagset.

HELP
provides generic help and information with a quick reference.

SETTINGS
provides the current option settings.

Requirement: All values must be enclosed in quotation marks.

Featured in: Example 2 on page 299

(SECT='*rtf_control_string*' | 'OFF' | 'NONE')
inserts RTF control words into the section data specifications.

rtf_control_string specifies RTF control words used to format the section data.

OFF turns off the usage of RTF control words for the section data and resets the *rtf_control_string* to null.

Alias: NO

NONE stops new RTF control words from being inserted into the file for the section data. ODS continues to use the section data information that was set before the use of NONE until it is reset.

Requirement: All values must be enclosed in quotation marks.

Tip: To reset the *rtf_control_string*, assign a different value or use the OFF or NO values.

See: Rich Text Format (RTF) Specification, version 1.6 available on the **MSDN** home Web page for information on RTF control words. Simply search for the document.

(TABLES_OFF='*style_elements*' | 'STYLE_ELEMENTS' | 'OFF')
determines whether tables will be used. A table can consist of one cell or many cells. SAS puts all of the text that you create into tables for RTF output. Use this suboption for tables that are text holders like titles, footnotes, and TEXT=. You should not use this suboption for tables produced by reporting procedures.

Note: To view the gridlines of tables in Microsoft Word, select **Show Gridlines** from the **Table** drop-down menu. △

style_elements
specifies the style element for formatting. For example, the following statement turns off tables that use the USERTEXT style element. The text specified by the TEXT= option is not placed in the table..

```
ods tagsets.rtf options (Tables_OFF='usertext');
ods tagsets.rtf text=''Text is not placed in a table'');
```

STYLE_ELEMENTS
lists the output style elements in the SAS log.

OFF
turns the option off. Therefore, ODS places the information output next into the RTF file inside a table. This action is the default option.

Alias: NO

Requirement: You must enclose all values in quotation marks.

See: "General ODS Style Elements" on page 905 for information about style elements

Featured in: Example 3 on page 300

(TOC_DATA ='ON' | 'OFF')
specifies whether to show the contents data in the RTF file.

OFF — instructs ODS not to display the table of contents data in the RTF file.
Alias: NO

ON — instructs ODS to display the hidden text of the table of contents in the RTF file.
Alias: YES

Requirement: You must enclose all values in quotation marks.

Featured in: Example 1 on page 295

(TROWD='*rtf_control_string*' | 'OFF')
inserts raw RTF specifications directly into header descriptions of the table row.

rtf_control_string specifies RTF control words and symbols.

OFF — RTF controls are no longer inserted.
Alias: NO

Requirement: You must enclose all values in quotation marks.

See: Rich Text Format (RTF) Specification, version 1.6 available on the **MSDN** home Web page for information on RTF control words. Search for the RTF 1.6 document.

Featured in: Example 4 on page 302

(TRHDR='*rtf_control_string*' | 'OFF')
inserts raw tablerow RTF specifications directly into the header description of the table row.

rtf_control_string specifies Microsoft RTF control words or symbols.

OFF — RTF controls are no longer inserted.
Alias: NO

Requirement: You must enclose all values in quotation marks.

See: Rich Text Format (RTF) Specification, version 1.6 available on the **MSDN** home Web page for information on RTF control words. Search for the RTF 1.6 document.

Featured in: Example 4 on page 302

(TROWHDRCELL='*text_string*' | 'OFF')
inserts raw text into the table row cells. If the RTF Reader does not recognize this *text_string*, it applies the raw text to the location where the RTF is being written in the documentation. Otherwise, the RTF Reader interprets the *text_string* as RTF control words.

text_string any text specified.

OFF — inserts a null string. Text is no longer inserted.
Alias: NO

Requirement: You must enclose all values in quotation marks.

See: Rich Text Format (RTF) Specification, version 1.6 available on the **MSDN** home Web page for information on RTF control words. Search for the RTF 1.6 document.

Featured in: Example 4 on page 302

Requirement: The OPTION suboption's must be enclosed in parentheses.

PAGEPANELS= *n* | NONE
specifies the number of panels permitted per page before ODS inserts a page break.

n
specifies a positive integer.

Default: 0

Tip: Setting the value to 0 resets the action to the default action.

NONE
specifies that paneling will be handled the way that it has always been handled by traditional ODS RTF. That is, all of the first panel is written, then all of the second panel, and so on, until all of the table information is written.

Default: If you do not specify paneling, ODS tries to fit the full set of panels on a single page. ODS measures the width of the text and tables (horizontal measurement) and determines what the column widths should be. ODS then divides the page into panels if it is too wide to fit on a page.

ODS always determines the column widths and determines whether panels are required. When there are multiple panels, ODS attempts to place a reasonable number of rows in each panel.

Featured in: Example 5 on page 304

TABLEROWS= *n*
specifies the number of rows in each table before ODS inserts a page break. If the table is narrow enough to fit on a page, *n* lines will be written to the table before a page break. If the table is too wide for a page, the page is broken into panels. In each panel, *n* rows will be written. When all the panels for *n* rows have been written, a page break is inserted before the next group of panels is written.

Note: Page breaks are not forced between panels. △

n
is a positive integer.

Alias: 0 | NONE

Default: Allow SAS to determine the number of rows per table.

Tip: 0 or NONE returns to the default, which allows SAS to determine the number of rows per table.

Featured in: Example 5 on page 304

UNIFORM
ensures uniformity from page to page within a single table that requires multiple pages. When the UNIFORM option is in effect, ODS reads the entire table before it starts to print it and determines the column widths that are necessary to accommodate all of the data. ODS applies these column widths to all pages of a multiple page table.

Note: With BY-group processing, SAS writes the results of each BY group to a separate table, so the output might not be uniform across BY groups. △

Default: If you do not specify the UNIFORM option, ODS prints a table one page at a time. This approach ensures that SAS does not run out of memory while it

processes very large tables. However, column widths might vary from one page to the next.

Tip: After this option is turned on, you cannot turn it off for that SAS session.

Tip: The UNIFORM option can cause SAS to run out of memory if you are printing a very large table. If this happens, you can specify the width of each of the columns in the table. Then print the table one page at a time. To do so, you must edit the table definition that you use. For more information, see "What You Can Do With a Table Template" on page 594.

Featured in: Example 6 on page 308

Details

Opening and Closing the ODS TAGSETS.RTF Destination You can modify and open an RTF destination with many ODS TAGSETS.RTF options. However, the FILE= option automatically closes the open destination that is referred to by the ODS TAGSETS.RTF statement. The option also closes any files associated with it and opens a new instance of the destination. If you use one of the ODS TAGSETS.RTF options, you should close the destination yourself.

Understanding How Traditional RTF Formats Output RTF produces output for Microsoft Word 2002. Although other applications can read RTF files, the RTF output might not work successfully with them.

The RTF destination enables you to view and edit the RTF output. ODS does not define the "vertical measurement," which means that SAS does not determine the optimal place to position each item on the page. For example, page breaks are not always fixed because you do not want your RTF output tables to split at inappropriate places. Your tables can remain intact on one page, or can have logical breaks where you specify.

Microsoft Word needs to know the widths of table columns, and it cannot adjust tables if they are too wide for the page. However, ODS measures the width of the text and tables (horizontal measurement). Therefore, all the column widths can be set properly by SAS, and the table can be divided into panels if it is too wide to fit on a single page.

In short, when producing RTF output for input to Microsoft Word, SAS determines the horizontal measurement and Microsoft Word controls the vertical measurement. Because Microsoft Word can determine how much room there is on the page, your tables will display consistently even after you modify your RTF file.

Note: The creation of complex tables that contain a large number of observations can reduce system efficiencies and increase processing time. △

ODS Measured RTF Versus Traditional ODS RTF The ODS RTF tagset (ODS TAGSETS.RTF), which is also referred to as the measured tagset, is new for SAS 9.2. This tagset enables users to specify how and where page breaks occur and when to place titles and footnotes into the body of a page. Traditional ODS RTF relies on Microsoft Word to make implicit page breaks for tables that are too long to fit on a single page. Traditional RTF also places titles and footnotes in the RTF instructions that enable Microsoft Word to apply them to pages as they are needed. In contrast, the RTF tagset enables SAS to place titles and footnotes into the body of the document so that it is outside of the control of Microsoft Word. Therefore, SAS becomes responsible for the implicit page breaks.

RTF Tagset Features The new "measured" RTF tagset does the following:

- controls page breaks on very large tables
- supports RTF readers other than Microsoft Word
- controls titles, footnotes, and other page elements

Controlling Page Breaks in Long Tables

Multiple-page tables can be a problem for ODS RTF. Like the ODS PRINTER destinations, SAS determines where to wrap a wide table. But for a long table, the entire table is loaded into memory before being rendered. When tables become longer than a physical page, Microsoft Word determines the page break. Microsoft word re-creates the column heading information in the table and applies titles and footnotes as needed. If a table is later edited in Microsoft Word, the information remains valid.

Unfortunately, a lot of information is associated with each cell of a table. No matter how much memory is added to the system, it is possible to create a table that can exceed it. Furthermore, an exhausted memory condition cannot be anticipated because it varies with the machine setup and with the table that you are creating.

However, with the ODS RTF tagset, SAS determines where to break the page and puts the titles and footnotes in the body of the document. When the table is broken into pages and SAS controls the page breaks, approximately a page of data is needed in memory at any one time. Therefore, a much smaller memory footprint is consumed and extremely large tables can be created. The ODS RTF tagset accommodates users who need large tables and users who want the old style RTF behavior. Both RTF implementations can be supported simultaneously.

Supporting RTF Readers Other than Word

Before SAS version 9.2, the traditional RTF architecture supported only the Microsoft Word RTF. The problem with supporting multiple readers is that RTF readers interpret the RTF specification in different ways. Now with the RTF tagset, you can enable subtle changes in one reader without impacting another RTF reader.

Controlling Titles, Footnotes, and Other Page Elements

Measured RTF uses a tagset that places the titles and footnotes on the page as tables instead of as RTF control words that are passed to Microsoft Word. With traditional RTF, the titles and footnotes are placed in the RTF header and footer information unless you specify the BODYTITLE option. Because the headers and footers are automatically placed in the body of the document with measured RTF, the TAGSET.RTF destination does not need the BODYTITLE option.

Measured RTF and Graphics

Measured RTF produces output in rich text format, which supports three formats for graphics that MS Word can read.

Format for Graphics	Corresponding SAS Graphics Driver
emfblips	SASEMF
pngblips	PNG
jpegblips	JPEG

When you do not specify a target device, the default target is PNG. You can also use the ACTIVEX, ACTXIMG, JAVAIMG graphics drivers to generate graphics in your measured RTF documents. The ACTIVEX driver generates an ActiveX control. The ACTXIMG and JAVAIMG drivers generate PNG files with the ACTIVEX Control or JAVA Applets appropriately. For more information about graphics devices, see *SAS/GRAPH: Reference*.

Note: When you specify the JAVA device in the ODS TAGSET.RTF statement, the JAVAIMG driver is used. △

Note: You cannot use UTF-8 encoding with the ACTIVEX device in RTF. When UTF-8 encoding is used, the ACTXIMG (activex image) device is used. △

Examples

Example 1: Creating a Table of Contents

ODS features:

ODS TAGSETS.RTF statement:

Action:

CLOSE

Options:

CONTENTS

TOC_DATA

Other SAS features:

OPTIONS statement

PROC FORMAT

PROC PRINT

PROC SORT

PROC REPORT

PROC TABULATE

Data set:

See “Creating the Grain_Production Data Set” on page 878.

Format:

See “Creating the $CNTRY Format” on page 869.

Program Description The following example creates a table of contents page that contains embedded table of contents data for some procedures, but not for others. The insertion of the table of contents data can be turned on and off in the middle of a program.

Program

Sort the data set Grain_Production. PROC SORT sorts the data, first by values of the variable Year, then by values of the variable Country, and finally by values of the variable Type.

```
proc sort data=Grain_Production;
   by year country type;
run;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create RTF output and create a new body file for each page of output. The ODS TAGSETS.RTF statement opens the RTF destination and creates RTF output. The CONTENTS suboption creates a table of contents page that contains a table of contents field that puts all of the contents information that is embedded in the document into a table of contents. This action occurs only if the TOC_DATA suboption is specified along with the CONTENTS suboption. The table of contents information is not embedded by default into the RTF file. You can turn on the insertion of the TOC data by specifying TOC_DATA='YES' or instruct ODS to not nsert this information by specifying TOC_DATA ='NO'.

```
ods tagsets.rtf file='Grain_Tagset.rtf' options(contents='yes' toc_data='yes');
```

Suppress the default BY line and specify a new value into the BY line. The NOBYLINE option suppresses the default BY line variable. The #BYVAL parameter specification inserts the current value of the BY variable Year into the title.

```
options nobyline;
title 'Leading Grain-Producing Countries';
title2 'for #byval(year)';
```

Produce a report. This PROC REPORT step produces a report on grain production. Each BY group produces a page of output. ODS creates a new body file for each BY group. The NOWINDOWS option instructs ODS to run PROC REPORT without the REPORT window and sends its output to the open output destinations.

```
proc report data=Grain_Production nowindows;
   by year;
   column country type kilotons;
   define country  / group width=14 format=$cntry.;
   define type     / group 'Type of Grain';
   define kilotons / format=comma12.;
   footnote 'Measurements are in metric tons.';
run;
```

Restore the default BY line and clear the second TITLE statement. The BYLINE option restores the default BY line. The TITLE2 statement clears the second TITLE statement.

```
options byline;
title2;
```

Suppress the insertion of table of contents data into the RTF file. The TOC_DATA='NO' option instructs ODS not to insert the table of contents data into the RTF file. Therefore, because the TABULATE procedure follows the TOC_DATA='NO' option, there will be no entry for the TABULATE procedure in the table of contents page.

```
ods tagsets.rtf options(toc_data='no');
```

Produce a report. The TABLE statement in the PROC TABULATE step uses three dimensions. Year defines pages, Country and Type define the rows, and Kilotons defines the columns. Therefore, PROC TABULATE explicitly produces one page of output for 1995 and one page for 1996 based on the years specified in the Grain_Production dead set. ODS also starts a new body file for each page.

```
proc tabulate data=Grain_Production format=comma12.;
   class year country type;
   var kilotons;
   table year,
         country*type,
         kilotons*sum=' ' / box=_page_ misstext='No data';
   format country $cntry.;
   footnote 'Measurements are in metric tons.';
run;
```

Enable the insertion of table of contents data into the RTF file. The TOC_DATA='YES' option instructs ODS to insert the table of contents data into the RTF file. There will be an entry for the PRINT procedure in the table of contents page when the PROC PRINT statement is executed.

```
ods tagsets.rtf options(toc_data='yes');
```

Print the Grain_Production DATA set.

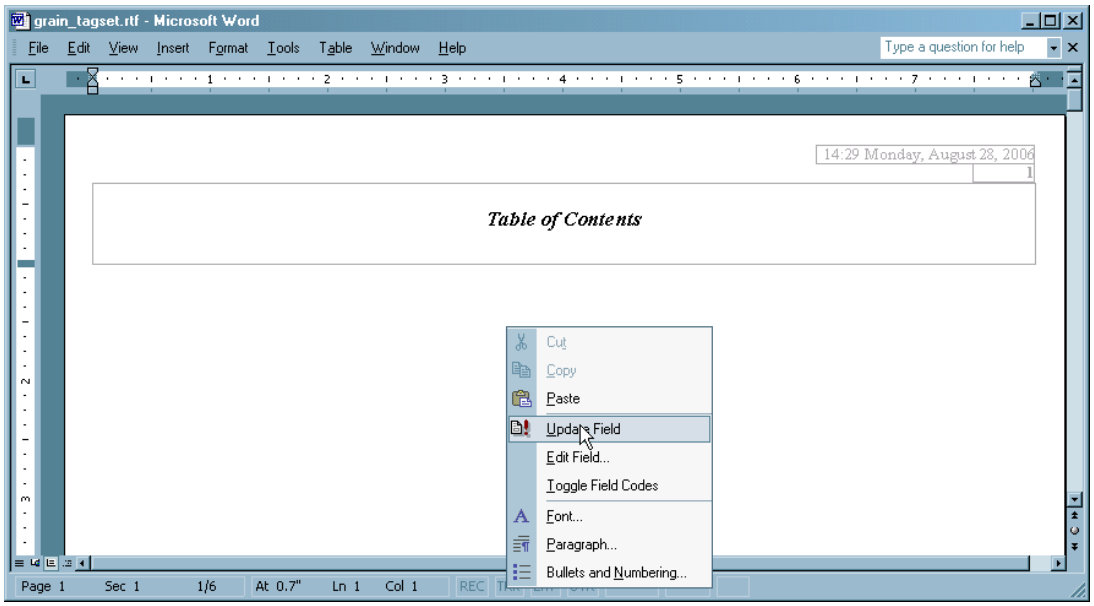
```
proc print data=Grain_Production;
run;
```

Close the RTF destination. The ODS TAGSETS.RTF CLOSE statement closes the RTF destination and all the files that are associated with it. If you do not close the destination, then you cannot view the files in a browser window.

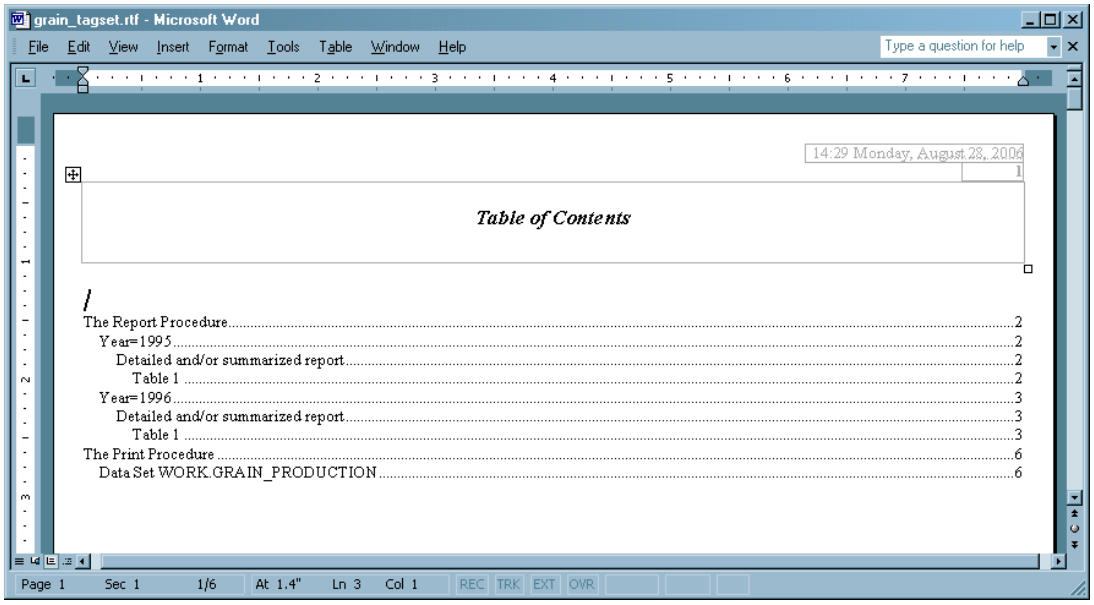
```
ods tagsets.rtf close;
```

RTF Output

By default the table of contents is collapsed on the table of contents page. To expand the table of contents from Microsoft Word, right-click beneath the title until the "Update Field" option is shown in the selection list. Then select “Update Field”.



The table of contents contains entries for PROC REPORT and PROC PRINT only. By default, ODS does not embed the table of contents data in the RTF document until you specify the TOC_DATA=’YES’ option, which results in an entry for PROC REPORT and all other data. If you turn off the TOC_DATA option before the TABULATE procedure, ODS does not insert information into the RTF document for PROC TABULATE. No other contents information is inserted into the RTF document until you specify TOC_DATA=’YES’. In this example, the TOC_DATA=’YES’ option is specified before the PRINT procedure. Therefore, ODS inserts contents data for PROC PRINT into the table of contents.



Example 2: Using the DOC Suboption to Get ODS TAGSETS.RTF Information

ODS features:
    ODS TAGSETS.RTF statement:
        Action:
            CLOSE
        Options:
            OPTIONS
                (DOC=”HELP”)
            FILE=

Other SAS features:
    PROC PRINT

Program Description The following example prints to the SAS log the OPTIONS suboptions and a description of each available suboption.

Program

Print information about the OPTIONS suboptions to the SAS log file. Specifying the OPTIONS suboption (DOC='HELP') prints Help for the ODS TAGSETS.RTF statement suboptions to the SAS log file. The FILE= option prints the data results to an RTF file named Help.rtf.

```
ods tagsets.rtf file='Help.rtf' options (doc="help");
```

Print the data set SASHELP.CLASS. The PROC PRINT statement prints the SASHELP.CLASS data set.

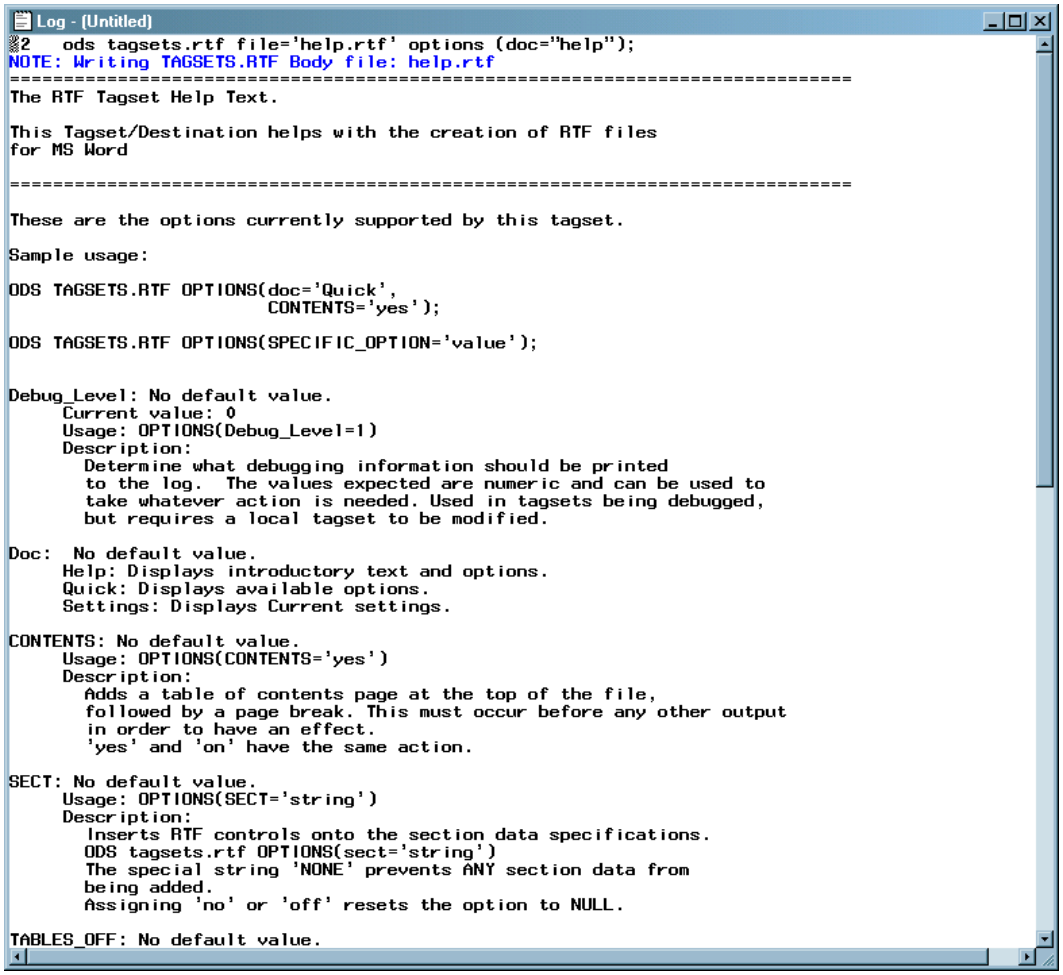
```
proc print data=Sashelp.Class;
run;
```

Close all destinations. Close the ODS TAGSETS.RTF destination and any other open destinations. This statement also closes all the files that are associated with each open destination. If you do not close a destination, then you cannot view the files in a browser window.

```
ods _all_ close;
```

RTF Output

Specify the "DOC='help' suboption to print all of the OPTIONS suboptions and information about each of the suboptions to the SAS log.



Example 3: Using the TABLES_OFF Suboption

ODS features:

ODS TAGSETS.RTF statement:

Action:

CLOSE

Options:

OPTIONS

(TABLES_OFF="OFF")

(TABLES_OFF="USERTEXT")

(TABLES_OFF="STYLE_ELEMENTS")

FILE=

TEXT=

Other SAS features:

PROC PRINT

Program Description The following example turns on and off the tables in RTF output and applies the style element specified by the TABLES_OFF suboption.

Program

List the style elements that can be applied in the SAS log file ODS TAGSETS.RTF enaables you to apply a style element to the RTF output. To determine the style elements that you can use, list them by specifying the TABLES_OFF suboption. This information is output to the SAS Log. Notice that you can use different style elements with each statement.

```
ods tagsets.rtf file="tablesOff.rtf" options(TABLES_OFF='STYLE_ELEMENTS');
proc print data=sashelp.class(obs=1) ;
run;
ods tagsets.rtf text="TEXT is placed in a table by default" ;
```

Turn off tables and apply the USERTEXT style element. Specifying TABLES_OFF=”USERTEXT”, applies the USERTEXT style to the text being output.

```
ods tagsets.rtf options(TABLES_OFF='usertext' );
ods tagsets.rtf text="TEXT is not placed in a table (table is removed when
style element is specified)" ;
```

Return to the default — tables are on. Specifying TABLES_OFF =”OFF”, returns the option to the default and turns the tables back on.

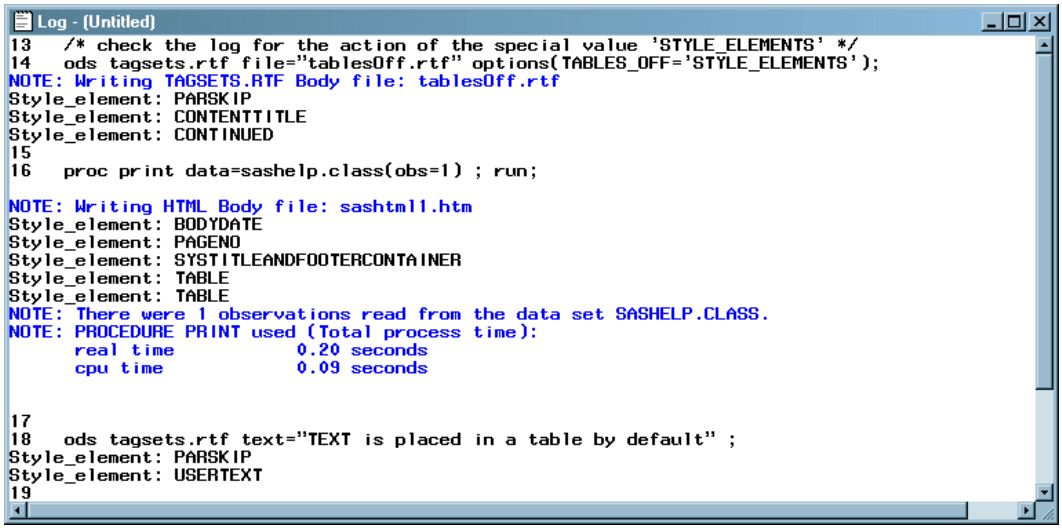
```
ods tagsets.rtf options(TABLES_OFF='off' );
ods tagsets.rtf text="TEXT is placed in a table (returned to default when
tables_off is set to off)" ;
```

Close all destinations. Close the ODS TAGSETS.RTF destination and any other open destinations. This statement also closes all of the files that are associated with each open destination. If you do not close a destination, you cannot view the files in a browser window.

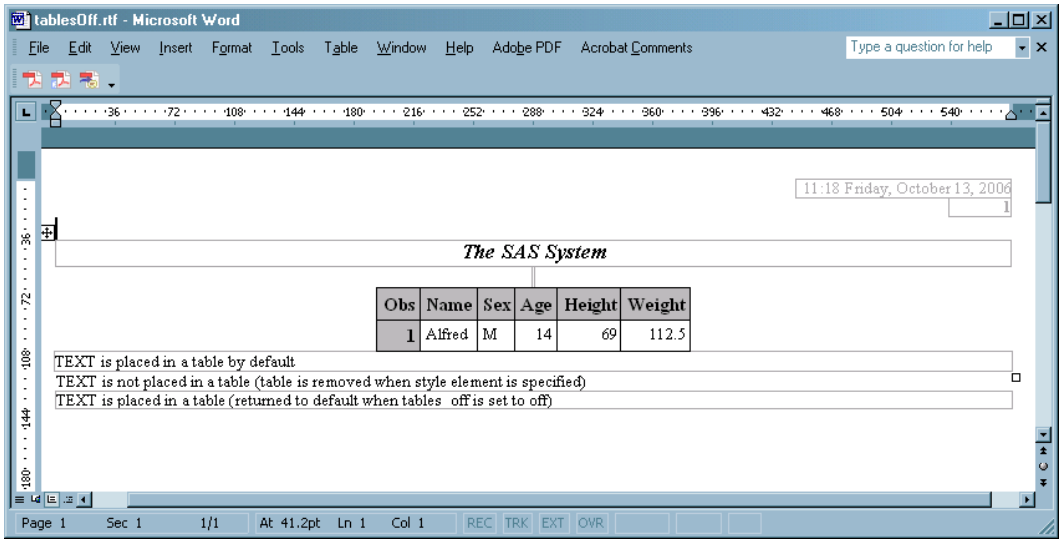
```
ods _all_ close;
```

RTF Output

If you specify the ODS TAGSETS.RTF suboption, TABLES_OFF= **style_element** lists the style elements that are being used and are output to the SAS log.



The following output illustrates what happens when the TABLES_OFF suboption is used. In this example, ODS places the output text in a table by default. Specifying TABLES_OFF=”USERTEXT” turns off the table and applies the USERTEXT style to the output. Lastly, TABLES_OFF=’OFF’ is specified, which causes the text to be output in a table.



Example 4: Column Heading Rotation Using the TRHDR, TROWHDRCELL, and TROWD Options

ODS features:

ODS TAGSETS.RTF statement:

Action:

CLOSE

Options:

OPTIONS

TRHDR=

TROWHDRCELL=

TROWD=

Other SAS features:

PROC PRINT

OPTIONS statement

Program Description The following example creates an RTF file in which the headers and contents of the row and column headings are rotated within the table.

Program

Specify the orientation of the page. The ORIENTATION option sets the page to landscape, the NODATE option turns off the output of the date and time, and the NONUMBER option tells SAS not to print the page number on the first title line of each page of output.

```
options orientation=landscape nodate nonumber;
```

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create RTF output using the ODS TAGSETS.RTF statement and rotate the rows and header information in the table. The ODS TAGSETS.RTF statement opens the RTF destination and creates RTF output that will be sent to the Mrotate.rtf file. The three options allow you to manipulate the row and header descriptions. TRHDR enables change to the table row headers. In this example, the RTF string that is specified adds more space to the row headers. TROWHDRCELL allows you to manipulate the table-row cell information. In this case, the information is rotated to vertical. The TROWD option allows you to change the table row description. The RTF string specified changes the first table row to the rightmost row.

```
ods tagsets.rtf file='Mrotate.rtf'
OPTIONS (TRHDR=''\trrh750''
         TROWHDRCELL=''\cltxbtlr''
         TROWD=''\rtlrow'');
```

Print the Sashelp.Class data set.

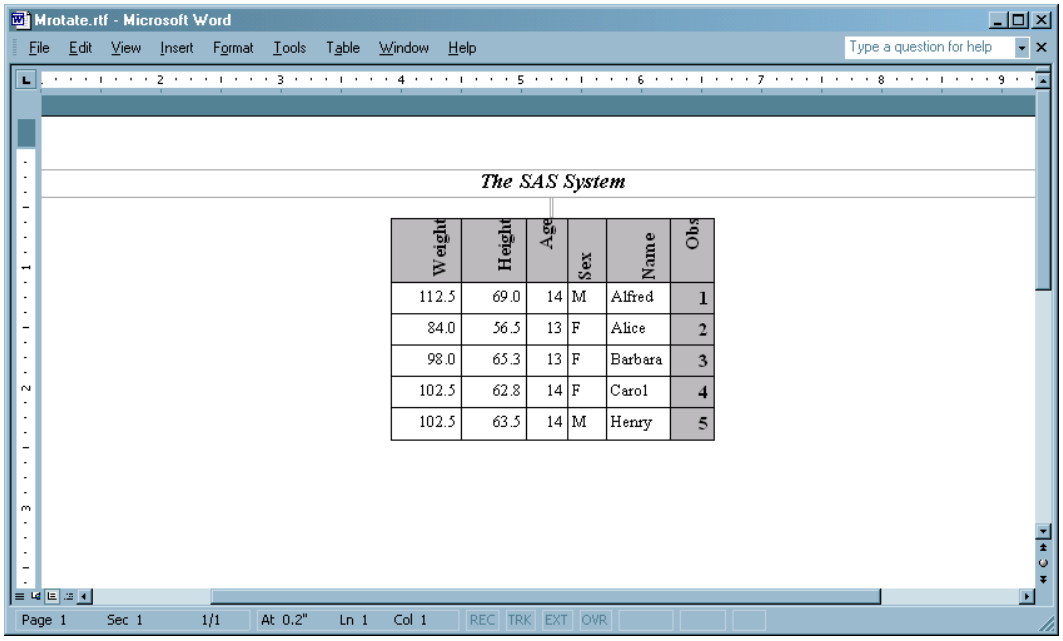
```
proc print data=Sashelp.Class(obs=5);
run;
```

Close the RTF destination. Close all destinations and all the files that are associated with each destination. If you do not close the destination, you cannot view the files in a browser window.

```
ods _all_ close;
```

RTF Output

The Mrotate.rtf output shows how ODS has rotated the first row of the table to the rightmost column, added more space to the row headers, and made the cell contents of the header row vertical. This table manipulation was caused by using the TRHDR=, TROWHDCELL=, and TROWD= suboptions of OPTIONS.



Example 5: Paneling Using the TABLEROWS and PAGEPANELS Options

ODS features:
- ODS TAGSETS.RTF statement:
  - Action:
    - CLOSE
  - Options:
    - TABLEROWS
    - PAGEPANELS

Other SAS features:
- OPTIONS statement
- PROC PRINT
- DATA statement

Program Description The following program provides various examples of how ODS creates panels when a table is wider than a page and presents some different choices for controlling the paneling.

Program

Specify the system options. The NODATE option turns off the output of the date and time, and the NONUMBER option tells SAS not to print the page number on the first title line of each page of output.

```
option nodate nonumber;
```

Close the LISTING destination so that SAS produces no listing output. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Open the RTF and file destination. Open the RTF destination and name the output file Panel.rtf. If you do not specify a file name, the output file name defaults to Sasmeas.rtf.

```
ods tagsets.rtf file=''Panel.rtf'';
```

Produce a large data set. Create a large data set in order to show how paneling works.

```
data temp;
array values val1--val50;
do j = 1 to 6;
    do i = 1 to dim(values);
        values(i) = i;
    end;
    output;
end;
run;
```

Create RTF output that uses the default paneling. The ODS TAGSETS.RTF statement opens the RTF destination and creates RTF output. Default paneling is used to print the TEMP data set that was created earlier in this program. The title of the table is "Default Paneling".

```
ods tagsets.rtf;
title ''Default Paneling'';
proc print data=Temp;
run;
```

Create RTF output where the number of panels is specified. The ODS TAGSETS.RTF statement opens the RTF destination and creates RTF output. RTF tagset options TABLEROWS and PAGEPANELS enable you to control the number of panels on a page and the number of rows of data that you want output for each table. The title of this multi-paneled table is "Paneling with TABLEROWS=5 and PAGEPANELS=4".

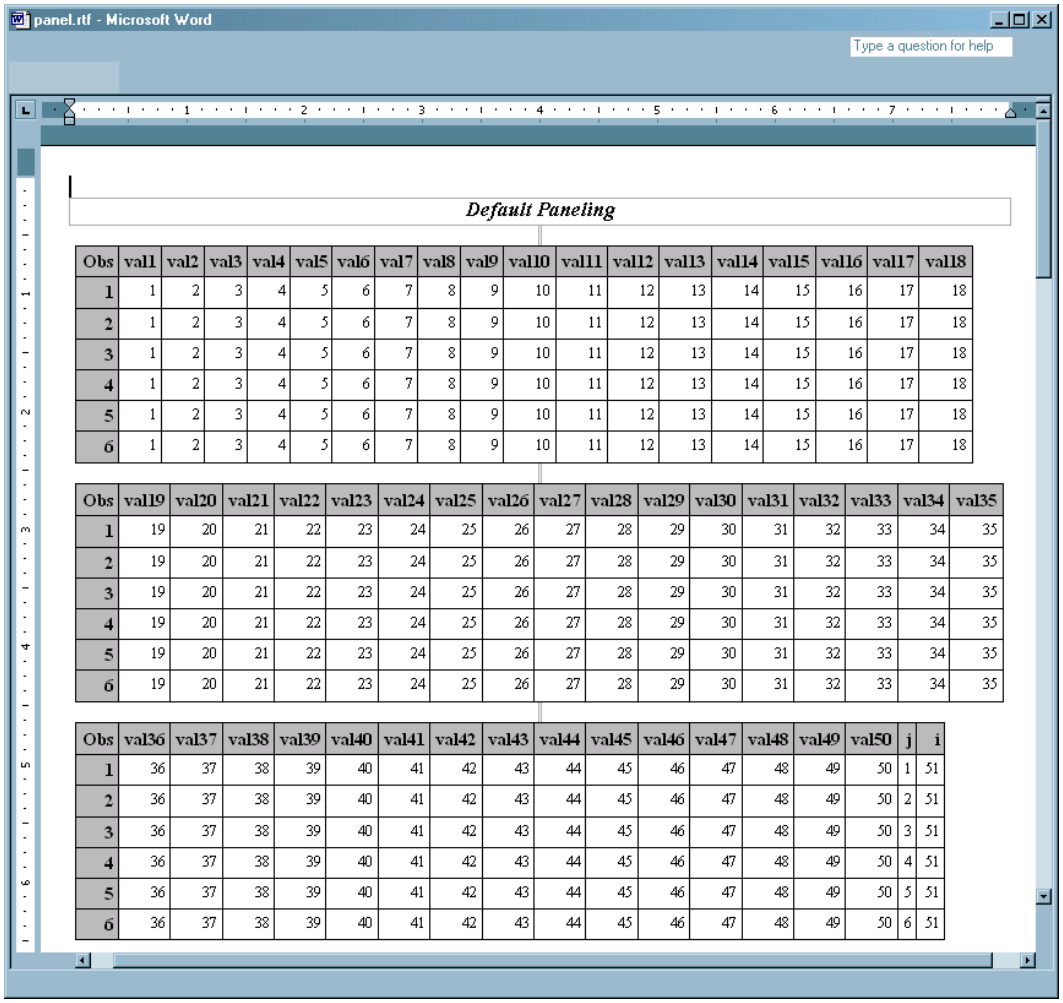
```
ods tagsets.rtf tablerows=5 pagepanels=4;
title 'Paneling with TABLEROWS=5 and PAGEPANELS=4';
proc print data=Temp;
run;
```

Close all destinations and restart the LISTING destination. The ODS TAGSETS.RTF CLOSE statement closes the open RTF destination and all of the files that are associated with it. If you do not close the destination, you cannot view the files in a browser window. The ODS LISTING statement opens the LISTING destination for output.

```
ods tagsets.rtf close;
ods listing;
```

RTF Output with Default Page Paneling

Page paneling occurs when a table is wider than a page. By default in measured ODS RTF, panels are grouped together so that all observations are close together. The first column holds as many columns as can fit on one line. The number of rows in each panel is determined by the number that fit on a logical page.



Default Paneling

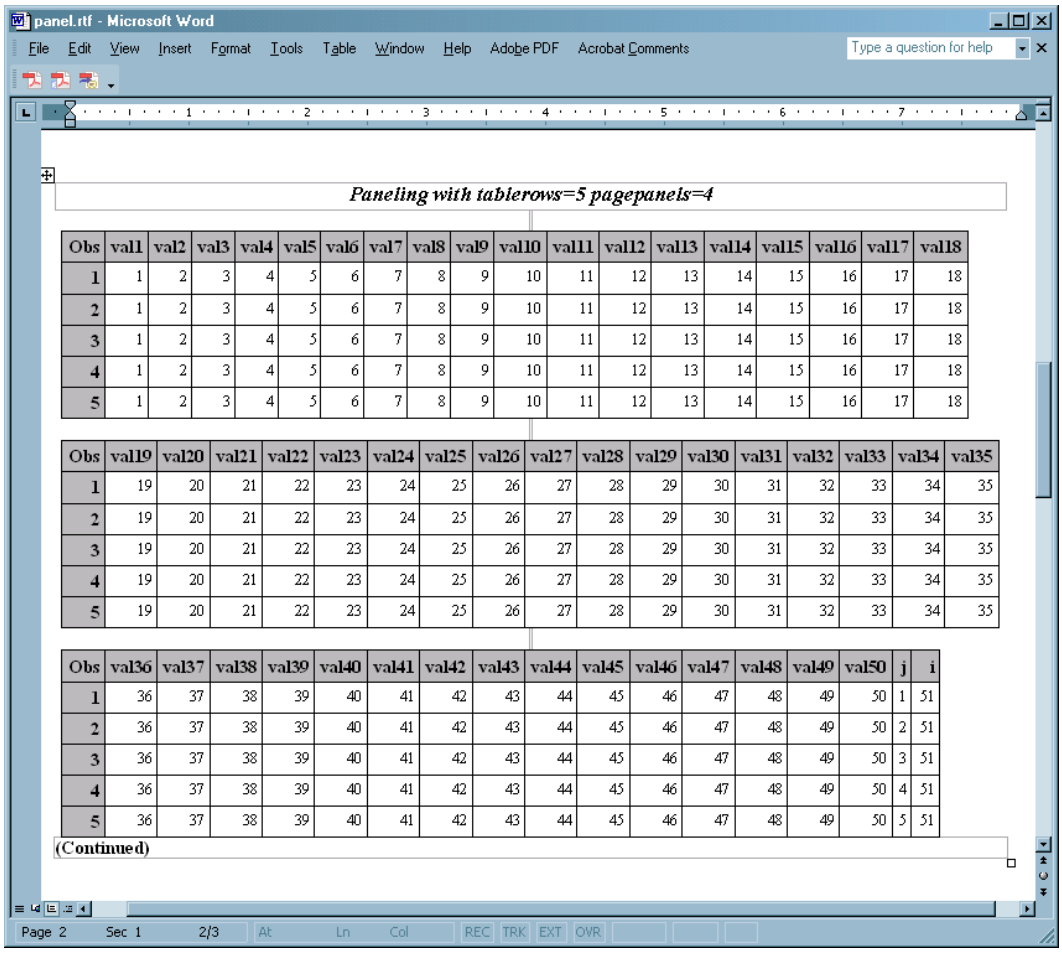
Obs	val1	val2	val3	val4	val5	val6	val7	val8	val9	val10	val11	val12	val13	val14	val15	val16	val17	val18
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Obs	val19	val20	val21	val22	val23	val24	val25	val26	val27	val28	val29	val30	val31	val32	val33	val34	val35
1	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
2	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
3	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
4	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
5	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
6	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Obs	val36	val37	val38	val39	val40	val41	val42	val43	val44	val45	val46	val47	val48	val49	val50	j	i
1	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	1	51
2	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	2	51
3	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	3	51
4	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	4	51
5	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	5	51
6	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	6	51

RTF Output with Options PAGEPANELS and TABLEROWS

The TABLEROWS option enables you to specify the number of rows for a panel. Note that only five rows are produced for each panel. The other row is presented on a separate page in three different panels.



panel.rtf - Microsoft Word

File Edit View Insert Format Tools Table Window Help Adobe PDF Acrobat Comments

Paneling with tablerows=5 pagepanels=4

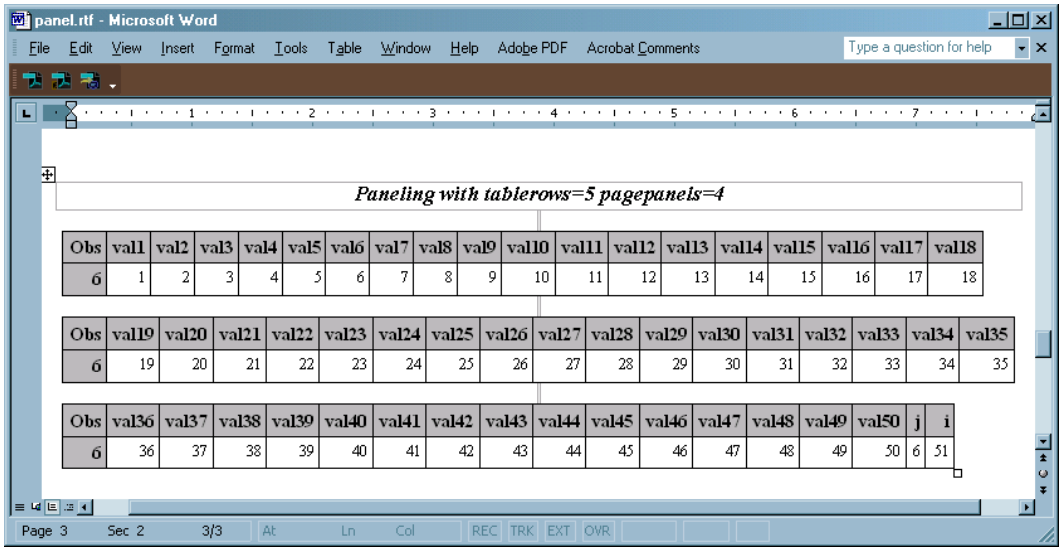
Obs	val1	val2	val3	val4	val5	val6	val7	val8	val9	val10	val11	val12	val13	val14	val15	val16	val17	val18
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Obs	val19	val20	val21	val22	val23	val24	val25	val26	val27	val28	val29	val30	val31	val32	val33	val34	val35
1	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
2	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
3	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
4	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
5	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Obs	val36	val37	val38	val39	val40	val41	val42	val43	val44	val45	val46	val47	val48	val49	val50	j	i
1	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	1	51
2	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	2	51
3	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	3	51
4	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	4	51
5	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	5	51

(Continued)

Page 2 Sec 1 2/3 At Ln Col REC TRK EXT OVR



panel.rtf - Microsoft Word

File Edit View Insert Format Tools Table Window Help Adobe PDF Acrobat Comments

Paneling with tablerows=5 pagepanels=4

Obs	val1	val2	val3	val4	val5	val6	val7	val8	val9	val10	val11	val12	val13	val14	val15	val16	val17	val18
6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Obs	val19	val20	val21	val22	val23	val24	val25	val26	val27	val28	val29	val30	val31	val32	val33	val34	val35
6	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Obs	val36	val37	val38	val39	val40	val41	val42	val43	val44	val45	val46	val47	val48	val49	val50	j	i
6	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	6	51

Page 3 Sec 2 3/3 At Ln Col REC TRK EXT OVR

Example 6: Repeating Headers Using the UNIFORM Option

ODS features:
- ODS TAGSETS.RTF statement:
  - Action:
    - CLOSE
  - Options:
    - UNIFORM
    - FILE=
- ODS RTF statement

Other SAS features:
- OPTIONS statement
- PROC FORMAT
- PROC TABULATE
- DATA statement

Program Description The following example creates a multi-page table that is uniform across several pages. The row and column heading labels are also carried over to each page.

Program

Specify the orientation of the page and name the RTF output. Specify landscape as the orientation of the page. Name the RTF output file to RtfTab.rtf.

```
options orientation=landscape;
ods rtf file=''RtfTab.rtf'';
```

Open the RTF file and create output that has UNIFORM header information The ODS TAGSETS.RTF statement opens the RTF file. The UNIFORM option ensures that the column headings and header information appear on each page.

```
ods tagsets.rtf file=''MrtfTab.rtf'' uniform;
```

Close the LISTING destination. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```
ods listing close;
```

Create the data set One. Create a data set that has five columns. Each column is composed of one to five subcolumns.

```
data one;
   do a=1 to 2;
      do b=1 to 2;
         do c=1 to 3;
            do d=1 to 3;
               do e=1 to 5;
                  output;
               end;
            end;
         end;
       end;
```

```
    end;
run;
```

Create user-defined formats PROC FORMAT creates the formats that SAS will use in the columns and subcolumns of the table.

```
proc format;
     value cars 1='DATSUN 200SX'
                2='PONTIAC FIERO';
     value colors 1='RED'
                  2='LIGHT BLUE'
                  3='YELLOW'
                  4='GREEN'
                  5='BROWN';
     value luxury 1='ALL THE WAY'
                  2='STANDARD OPTIONS'
                  3='STRIPPED DOWN';
     value opts 1='POWER STEERING'
                2='SUN ROOF'
                3='AUTOMATIC'
                4='T-TOP'
                5='HATCHBACK'
                6='FUEL-INJECTION'
                7='HUBCAPS'
                8='AM/FM STEREO'
                9='FLOOR MATS'
                10='CASSETTE PLAYER';
     value perform 1='VERY SLOW'
                   2='SLOW'
                   3='AVERAGE'
                   4='FAST'
                   5='VERY FAST';
  run;
```

Create data set Two. Data set Two populates the data set with the formats supplied by PROC FORMAT.

```
data two (keep=model color luxury options perform);
  length model color luxury options perform  $ 20;
     set one;
     model=put(a,cars.);
     color=put(b,colors.);
     luxury=put(c,luxury.);
     options=put(d,opts.);
     perform=put(e,perform.);
  run;
```

Create titles for the Output Provide two titles for the output.

```
title2 'My Favorite Cars';
  title3 '(large data set)';
```

Produce a report. PROC TABULATE creates the table of cars and their attributes.

```
proc tabulate data=two order=data ;
     class model color luxury options perform;
```

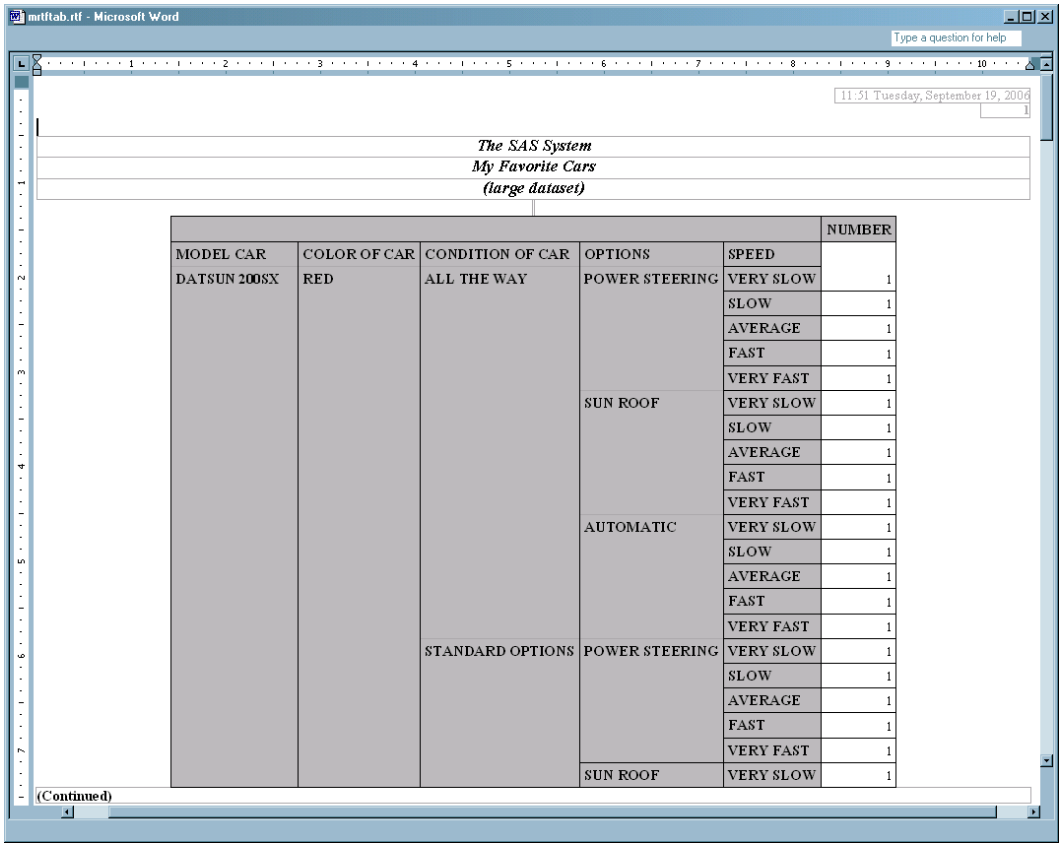
```
    table model*color*luxury*options*perform,n / indent=4 condense;
    label model='MODEL CAR'
          color='COLOR OF CAR'
          luxury='CONDITION OF CAR'
          perform='SPEED';
    keylabel n='NUMBER';
run;
```

Close all destinations. The ODS_ALL_CLOSE statement closes any open destinations and all of the files that are associated with them. If you do not close the destination, you cannot view the files in a browser window.

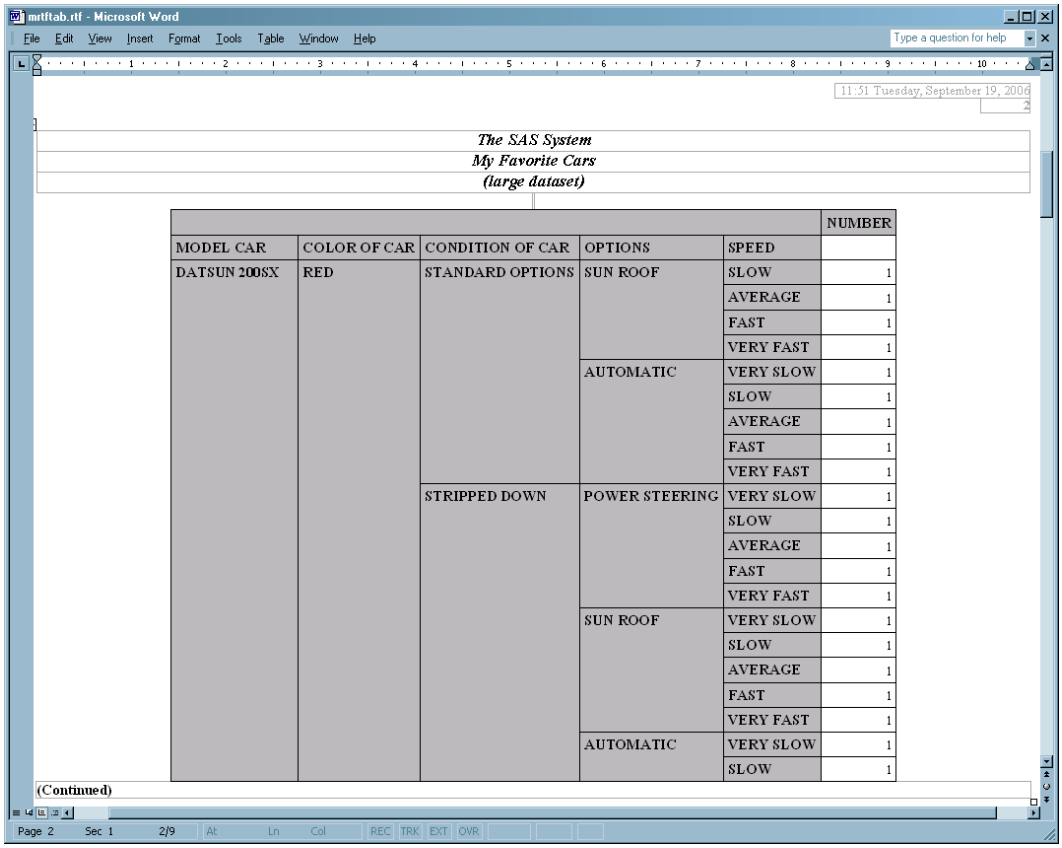
```
ods _all_ close;
```

Measured RTF Output

The following output is from the measured RTF output file Mrtftab.rtf. This output is generated using the ODS TAGSETS.RTF statement. Note the differences between the measured output and the traditional RTF output. Note that the cell header information is carried to each page and that the word "Continued" appears at the bottom of each page of RTF output.



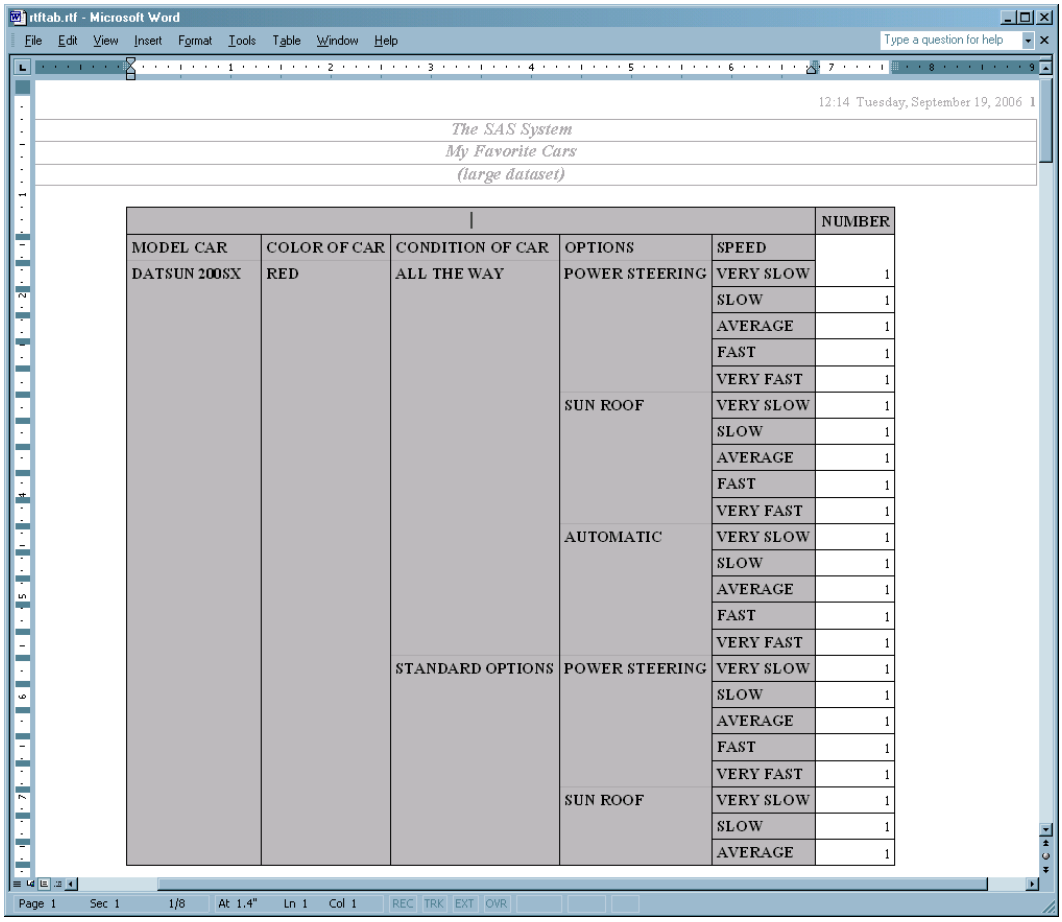
					NUMBER
MODEL CAR	COLOR OF CAR	CONDITION OF CAR	OPTIONS	SPEED	
DATSUN 200SX	RED	ALL THE WAY	POWER STEERING	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			SUN ROOF	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			AUTOMATIC	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
		STANDARD OPTIONS	POWER STEERING	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			SUN ROOF	VERY SLOW	1



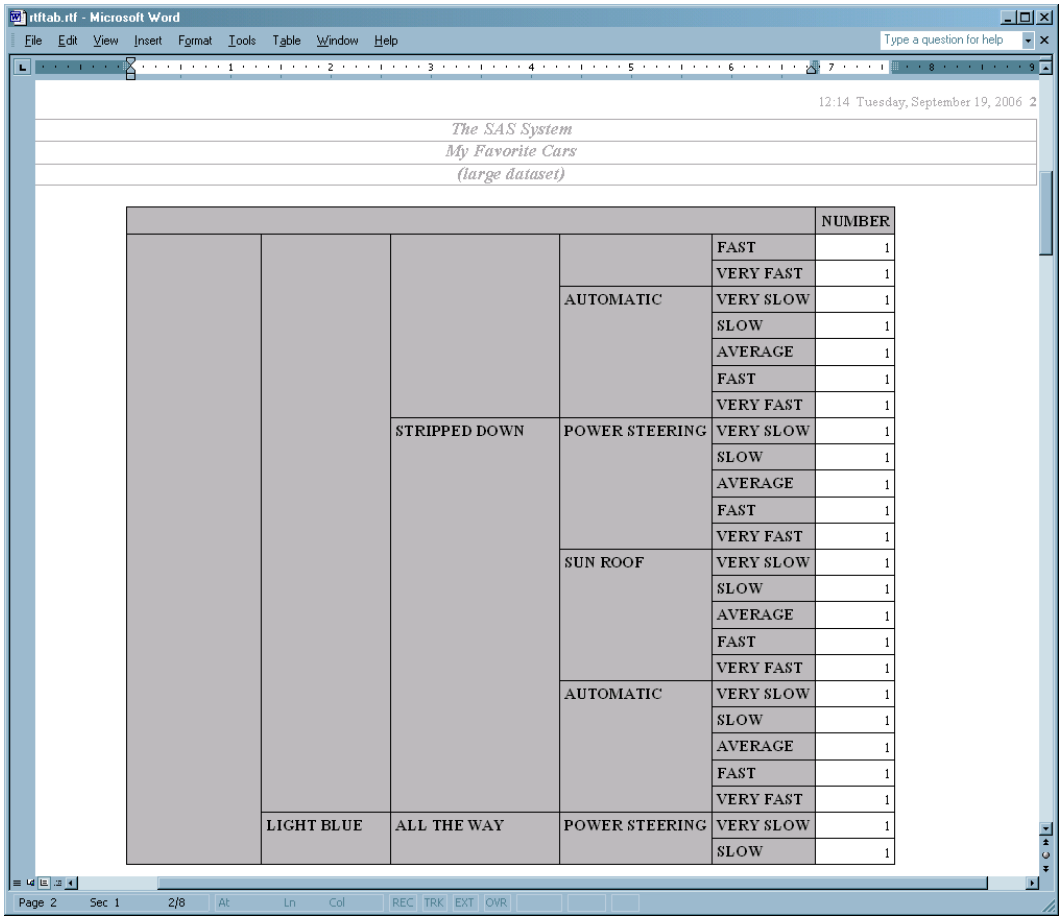
					NUMBER
MODEL CAR	COLOR OF CAR	CONDITION OF CAR	OPTIONS	SPEED	
DATSUN 200SX	RED	STANDARD OPTIONS	SUN ROOF	SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			AUTOMATIC	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
		STRIPPED DOWN	POWER STEERING	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			SUN ROOF	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			AUTOMATIC	VERY SLOW	1
				SLOW	1

Traditional RTF Output

The following output is a portion of the Rtftab.rtf file that was generated using the traditional ODS RTF statement. Notice that header information is not carried over to page two of the output. Also note that page one does not indicate that more pages of output follow.



					NUMBER
MODEL CAR	COLOR OF CAR	CONDITION OF CAR	OPTIONS	SPEED	
DATSUN 200SX	RED	ALL THE WAY	POWER STEERING	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			SUN ROOF	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			AUTOMATIC	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
		STANDARD OPTIONS	POWER STEERING	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			SUN ROOF	VERY SLOW	1
				SLOW	1
				AVERAGE	1



rtftab.rtf - Microsoft Word

File Edit View Insert Format Tools Table Window Help

Type a question for help

12:14 Tuesday, September 19, 2006 2

The SAS System
My Favorite Cars
(large dataset)

					NUMBER
				FAST	1
				VERY FAST	1
			AUTOMATIC	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
		STRIPPED DOWN	POWER STEERING	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			SUN ROOF	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
			AUTOMATIC	VERY SLOW	1
				SLOW	1
				AVERAGE	1
				FAST	1
				VERY FAST	1
	LIGHT BLUE	ALL THE WAY	POWER STEERING	VERY SLOW	1
				SLOW	1

Page 2 Sec 1 2/8 At Ln Col REC TRK EXT OVR

ODS TEXT= Statement

Inserts text into your ODS output.

Valid in: anywhere

Category: ODS: Output Control

Tip: The ODS TEXT= statement is sent only to output destinations that are open. Therefore, it must be specified after an ODS destination statement.

Syntax

ODS TEXT= '*text-string*'

Required Arguments

text-string

specifies the text to insert into your output. This text is sent to all open supported output destinations.

Restriction: The ODS TEXT= statement does not support the OUTPUT destination or the LISTING destination. All other ODS destinations are supported.

Requirement: You must enclose '*text-string*' in parentheses.

Tip: The UserText style element controls text specified with the TEXT= statement.

Examples

Example 1: Adding Text to Multiple Destinations

ODS features:

- ODS HTML statement
- ODS PDF statement
- ODS RTF statement
- ODS TEXT= statement
- PROC TEMPLATE:
  - DEFINE STYLE statement
  - PARENT= statement
  - STYLE statement

Other SAS features:

- PROC PRINT

Data set:

See "Creating the Exprev Data Set" on page 875.

Program Description The following example uses a single ODS TEXT= statement to add text to PDF, HTML, and traditional RTF output. PROC TEMPLATE modifies the UserText style element which controls the font style, font color, and other attributes of the text that the ODS TEXT= statement adds.

Program

```
options obs=10;
```

Create the MyStyle style template. The MyStyle style templates modifies the Usertext style element to change the font color of text created by the TEXT= statement to red.

```
proc template;
   define style mystyle;
   parent=styles.default;
   style usertext from usertext /
      foreground=red;
   end;
run;
```

Send output to multiple ODS destinations. The following statements open the HTML, PDF, and RTF destinations.

```
ods html file="text.html" style=mystyle;
ods pdf file="text.pdf" startpage=never notoc style=mystyle;
ods rtf file="text_trad.rtf" style=mystyle;
```

Add text strings before the ouput is printed. The ODS TEXT= statements add the text strings “My Text 1” and “My Text 2”. The text is added to the output before the data set is printed.

```
ods text="My Text 1";
ods text="My Text 2";
```

Add a title and footnote. The TITLE and FOOTNOTE statements specify the title and footnote.

```
title "January Orders ";
footnote " For All Employees9";
```

Print the data set Exprev. The PRINT procedure prints the Exprev data set.

```
proc print data=exprev;
run;
```

Add a third text string after the data set. The third ODS TEXT= statement adds the text string “My Text 3” after the data set is printed.

```
ods text="My Text 3";
```

Close all open destinations and remove the titles and footnotes. The ODS _ALL_ CLOSE statement closes all open ODS destinations. The TITLE and FOOTNOTE statements remove any titles and footnotes previously specified.

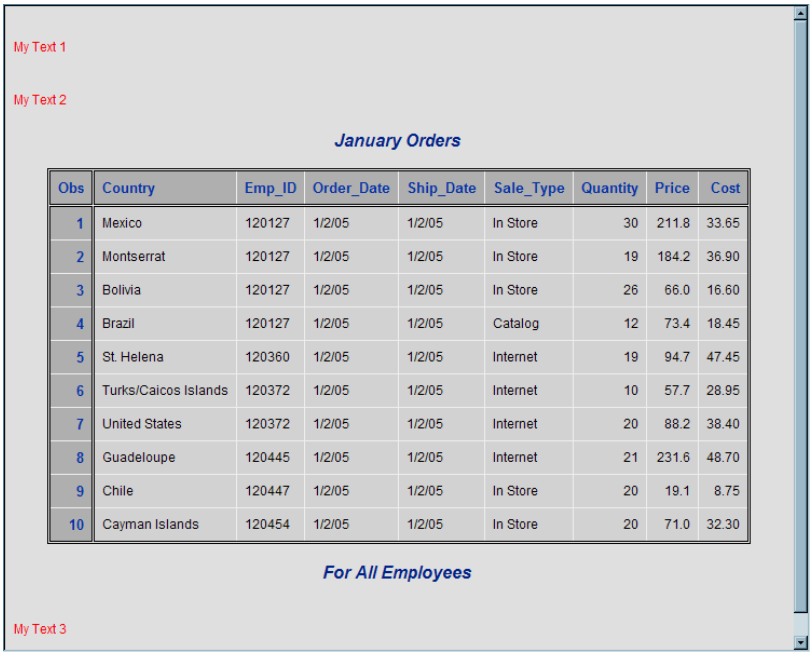
```
ods _all_ close;

title;
footnote;
```

Delete the MyStyle style template. The DELETE statement deletes the MyStyle style template.

```
proc template;
   delete mystyle;
   run;
```

Output

Display 5.25 HTML Output with Text Added



My Text 1

My Text 2

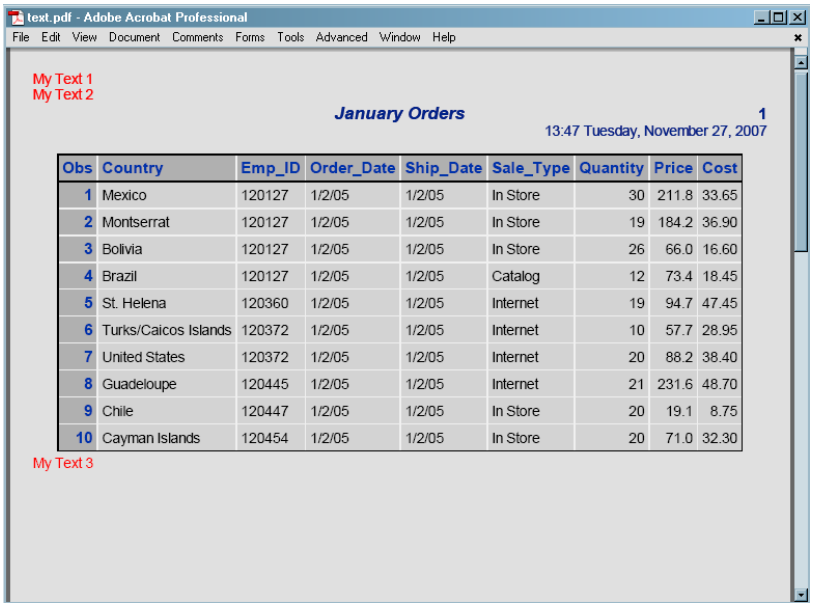
January Orders

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Mexico	120127	1/2/05	1/2/05	In Store	30	211.8	33.65
2	Montserrat	120127	1/2/05	1/2/05	In Store	19	184.2	36.90
3	Bolivia	120127	1/2/05	1/2/05	In Store	26	66.0	16.60
4	Brazil	120127	1/2/05	1/2/05	Catalog	12	73.4	18.45
5	St. Helena	120360	1/2/05	1/2/05	Internet	19	94.7	47.45
6	Turks/Caicos Islands	120372	1/2/05	1/2/05	Internet	10	57.7	28.95
7	United States	120372	1/2/05	1/2/05	Internet	20	88.2	38.40
8	Guadeloupe	120445	1/2/05	1/2/05	Internet	21	231.6	48.70
9	Chile	120447	1/2/05	1/2/05	In Store	20	19.1	8.75
10	Cayman Islands	120454	1/2/05	1/2/05	In Store	20	71.0	32.30

For All Employees

My Text 3

Display 5.26 PDF Output with Text Added



text.pdf - Adobe Acrobat Professional

File Edit View Document Comments Forms Tools Advanced Window Help

My Text 1
My Text 2

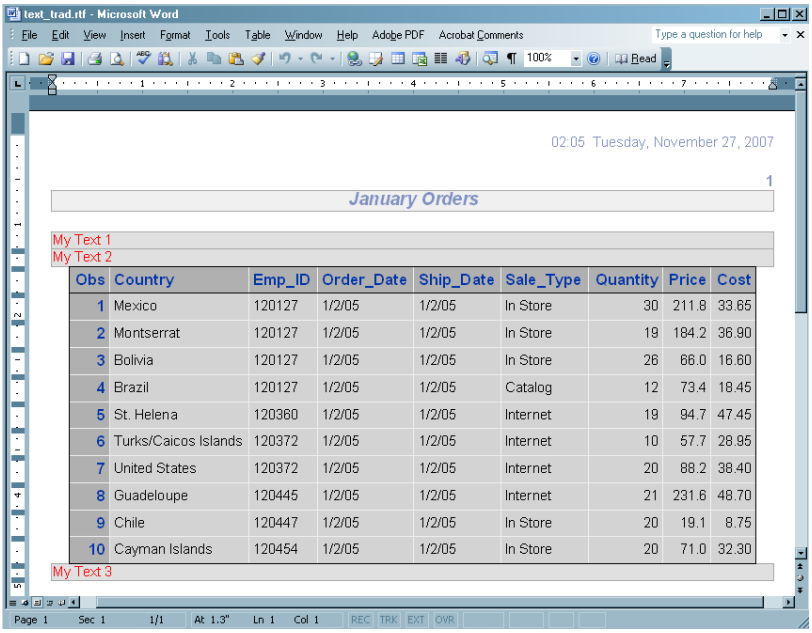
January Orders

1
13:47 Tuesday, November 27, 2007

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Mexico	120127	1/2/05	1/2/05	In Store	30	211.8	33.65
2	Montserrat	120127	1/2/05	1/2/05	In Store	19	184.2	36.90
3	Bolivia	120127	1/2/05	1/2/05	In Store	26	66.0	16.60
4	Brazil	120127	1/2/05	1/2/05	Catalog	12	73.4	18.45
5	St. Helena	120360	1/2/05	1/2/05	Internet	19	94.7	47.45
6	Turks/Caicos Islands	120372	1/2/05	1/2/05	Internet	10	57.7	28.95
7	United States	120372	1/2/05	1/2/05	Internet	20	88.2	38.40
8	Guadeloupe	120445	1/2/05	1/2/05	Internet	21	231.6	48.70
9	Chile	120447	1/2/05	1/2/05	In Store	20	19.1	8.75
10	Cayman Islands	120454	1/2/05	1/2/05	In Store	20	71.0	32.30

My Text 3

Display 5.27 Traditional RTF Output with Text Added



text_trad.rtf - Microsoft Word

02:05 Tuesday, November 27, 2007

1

January Orders

My Text 1
My Text 2

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Mexico	120127	1/2/05	1/2/05	In Store	30	211.8	33.65
2	Montserrat	120127	1/2/05	1/2/05	In Store	19	184.2	36.90
3	Bolivia	120127	1/2/05	1/2/05	In Store	26	66.0	16.60
4	Brazil	120127	1/2/05	1/2/05	Catalog	12	73.4	18.45
5	St. Helena	120360	1/2/05	1/2/05	Internet	19	94.7	47.45
6	Turks/Caicos Islands	120372	1/2/05	1/2/05	Internet	10	57.7	28.95
7	United States	120372	1/2/05	1/2/05	Internet	20	88.2	38.40
8	Guadeloupe	120445	1/2/05	1/2/05	Internet	21	231.6	48.70
9	Chile	120447	1/2/05	1/2/05	In Store	20	19.1	8.75
10	Cayman Islands	120454	1/2/05	1/2/05	In Store	20	71.0	32.30

My Text 3

ODS TRACE Statement

Writes to the SAS log a record of each output object that is created, or suppresses the writing of this record.

Valid: anywhere

Category: ODS: Output Control

Default: OFF

Featured in: Example 3 on page 194

Syntax

ODS TRACE ON<*/option(s)*>;

ODS TRACE OFF;

Required Arguments

OFF
turns off the writing of the trace record.

Alias: NO

ON
turns on the writing of the trace record.

Alias: OUTPUT

Alias: YES

Options

EXCLUDED
includes, in the trace record, information for excluded output objects.

Featured in: Example 2 on page 273

LABEL
includes the label path for the output object in the record. You can use a label path anywhere that you can use a path.

Tip: This option is helpful for users who are running a localized version of SAS, because the labels are translated from English to the local language. The names and paths of output objects are not translated because they are part of the syntax of the Output Delivery System.

LISTING
writes the trace record to the Listing destination, so that each part of the trace record immediately precedes the output object that it describes.

Details

Contents of the Trace Record ODS produces an output object by combining data from the data component with a table definition. The trace record provides information about the data component, the table definition, and the output object. By default, the record that the ODS TRACE statement produces contains these items:

Name
is the name of the output object. You can use the name to reference this output object and others with the same name. For details on how to reference an output object, see “How ODS Determines the Destinations for an Output Object” on page 35. For example, you could use this name in an ODS OUTPUT statement to make a data set from the output object, or you could use it in an ODS SELECT or an ODS EXCLUDE statement.

Tip: The name is the rightmost part of the path that appears in the trace record.

Label
briefly describes the contents of the output object. This label also identifies the output object in the Results window.

Data name
is the name of the data component that was used to create this output object. The data name appears only if it differs from the name of the output object.

Data label
describes the contents of the data.

Template
is the name of the table definition that ODS uses to format the output object. You can modify this definition with PROC TEMPLATE. See the “EDIT Statement” on page 597 for more information.

Path
is the path of the output object. You can use the path to reference this output object. For example, you could use the path in the ODS OUTPUT statement to

make a data set from the output, or you could use it in an ODS SELECT or an ODS EXCLUDE statement.

The LABEL option modifies the trace record by including the label path for the object in the record. See the discussion of the LABEL option.

Specifying an Output Object After you have determined which output objects your SAS program produces, you can specify the output objects in statements such as ODS EXCLUDE, ODS SELECT, and so on. You can specify an output object by using one of the following:

- a full path. For example,

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  is the full path of the output object.
- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  then the partial paths are:

  ```
  City_Pop_90.TestsForLocation
  TestsForLocation
  ```

- a label that is enclosed by quotation marks.

  For example,

  ```
  "The UNIVARIATE Procedure"
  ```

- a label path. For example, the label path for the output object is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```

  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement. △
- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```

  then the partial label paths are:

  ```
  "CityPop_90"."Tests For Location"
  "Tests For Location"
  ```

- a mixture of labels and paths.
- any of the partial path specifications, followed by a pound sign (#) and a number. For example, **`TestsForLocation#3`** refers to the third output object that is named **`TestsForLocation`**.

Example

Example 1: Determining Which Output Objects a Procedure Creates

ODS features:

ODS TRACE statement:
LABEL
OFF
ON

Other SAS features:
PROC UNIVARIATE

Data set:
See “Creating the StatePop Data Set” on page 881.

This example shows how to determine the names and labels of the output objects that a procedure creates. You can use this information to select and exclude output objects.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 903. △

Program

Specify that SAS write the trace record to the SAS log and include label paths. This ODS TRACE statement writes the trace record to the SAS log. The LABEL option includes label paths in the trace record.

```
ods trace on / label;
```

Create descriptive statistics for two variables. PROC UNIVARIATE computes descriptive statistics for two variables, CityPop_80 and CityPop_90. As PROC UNIVARIATE sends each output object to the Output Delivery System, ODS writes the pertinent information for that output object to the trace record.

```
proc univariate data=statepop mu0=3.5;
   var citypop_90 citypop_80;
run;
```

Specify that SAS stop writing the trace record. The ODS TRACE OFF statement stops the writing of the trace record to the SAS log.

```
ods trace off;
```

SAS Log

This partial SAS log shows the trace record that the ODS TRACE statement creates. For each analysis variable PROC UNIVARIATE creates five output objects : **Moments**, **BasicMeasures**, **TestsForLocation**, **Quantiles**, and **ExtremeObs**.

Notice that an output object has the same name and label, regardless of which variable is analyzed. Therefore, you can select all the moments tables that PROC UNIVARIATE produces by using the name or label in an ODS SELECT statement. The path and label path are unique for each output object because they include the name of the variable that is analyzed. You can, therefore, select an individual moments table by using the path or the label path in an ODS SELECT statement.

```
Output Added:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.CityPop_90.Moments
Label Path: "The Univariate Procedure"."CityPop_90"."Moments"
-------------

Output Added:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.CityPop_90.BasicMeasures
Label Path: "The Univariate Procedure"."CityPop_90"."Basic Measures of Location and Variability"
-------------

Output Added:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.CityPop_90.TestsForLocation
Label Path: "The Univariate Procedure"."CityPop_90"."Tests For Location"
-------------

Output Added:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.CityPop_90.Quantiles
Label Path: "The Univariate Procedure"."CityPop_90"."Quantiles"
-------------
Output Added:
-------------
Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.CityPop_90.ExtremeObs
Label Path: "The Univariate Procedure"."CityPop_90"."Extreme Observations"
-------------
```

```
Output Added:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.CityPop_80.Moments
Label Path: "The Univariate Procedure"."CityPop_80"."Moments"
-------------

Output Added:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.CityPop_80.BasicMeasures
Label Path: "The Univariate Procedure"."CityPop_80"."Basic Measures of Location and Variability"
-------------

Output Added:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.CityPop_80.TestsForLocation
Label Path: "The Univariate Procedure"."CityPop_80"."Tests For Location"
-------------

Output Added:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.CityPop_80.Quantiles
Label Path: "The Univariate Procedure"."CityPop_80"."Quantiles"
-------------

Output Added:
-------------
Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.CityPop_80.ExtremeObs
Label Path: "The Univariate Procedure"."CityPop_80"."Extreme Observations"
-------------
```

See Also

Statements:

"ODS EXCLUDE Statement" on page 110
"ODS SELECT Statement" on page 264

ODS USEGOPT Statement

Determines whether ODS uses traditional SAS/GRAPH option settings.

Valid: anywhere

Category: ODS: Output Control

See also: *SAS/GRAPH: Reference*

Restriction: The ODS USEGOPTS option has no effect on graphics produced as a result of any of the ODS graphics functionality or the ODS GRAPHICS statement.

Syntax

ODS USEGOPT | NOUSEGOPT;

ODS USEGOPT
specifies that ODS use traditional SAS/GRAPH option settings for non-graphical output.

ODS NOUSEGOPT
specifies that ODS not use traditional SAS/GRAPH option settings for non-graphical output.

Details

Enabling Traditional SAS/GRAPH Graphics Options While ODS USEGOPT is in effect, the settings for the following graphics options will affect all of your ODS output, including tables:

CTEXT=
CTITLE=
FTITLE=
FTEXT=
HTEXT=
HTITLE=

If ODS NOUSEGOPT is in effect, the settings for these graphics options will not override the value in the style definition for titles and footnotes in your ODS output.

Examples

Example 1: Enabling and Disabling Graphics Options

ODS features:
- ODS HTML statement:
  - FILE=
- ODS LISTING statement:
  - CLOSE
- ODS NOUSEGOPT statement
- ODS USEGOPT statement

Other SAS features:
- GOPTIONS statement:
  - FCTEXT=
  - FTITLE=
  - HTEXT=
- PROC PRINT
- TITLE statement

Data set:
- See “Creating the Exprev Data Set” on page 875.

Program Description This example creates two HTML reports, one with the GOPTIONS enabled by using the ODS USEGOPT statement, and one with GOPTIONS disabled by using the ODS NOUSEGOPT statement.

Program

Specify the GOPTIONS. The RESET=ALL option sets all graphics options to their default values and cancels all global statements. The HTEXT= option specifies that the text height for titles and footnotes be two units. The FTITLE= option specifies the font for titles and footnotes. The FTEXT option specifies the font for the text.

```
goptions reset=all  htext=2 ftitle=script ftext=script;
```

Do not produce listing output. The ODS LISTING statement closes the LISTING destination to conserve resources. Otherwise, output would be written to the LISTING destination by default.

```
ods listing close;
```

Enable the graphics options. While ODS USEGOPT is in effect, the settings for HTEXT= and CTEXT= graphics options will override values that are specified for titles and footnotes in the style definition.

```
ods usegopt;
```

Create HTML output, specify titles, and print the data set. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file specified by the FILE= option.

The TITLE statements specify the titles for your output.

The PRINT procedure prints the SAS data set Exprev. The OBS= option specifies two observations to be printed.

```
ods html file='opts.html';
title 'This Title Was Created With the USEGOPT Option Specified ' ;
title2 'The Graphics Option Settings are Turned On';
proc print data=exprev(obs=2);
run;
```

Disable the graphics options. The NOUSEGOPT statement suppresses the use of the HTEXT= and CTEXT= graphics option settings for your output.

```
ods nousegopt;
```

Create HTML output, specify titles, and print the data set. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file specified by the FILE= option.

The TITLE statements specify the titles for your output.

The PRINT procedure prints the SAS data set Exprev. The OBS= option specifies two observations to be printed.

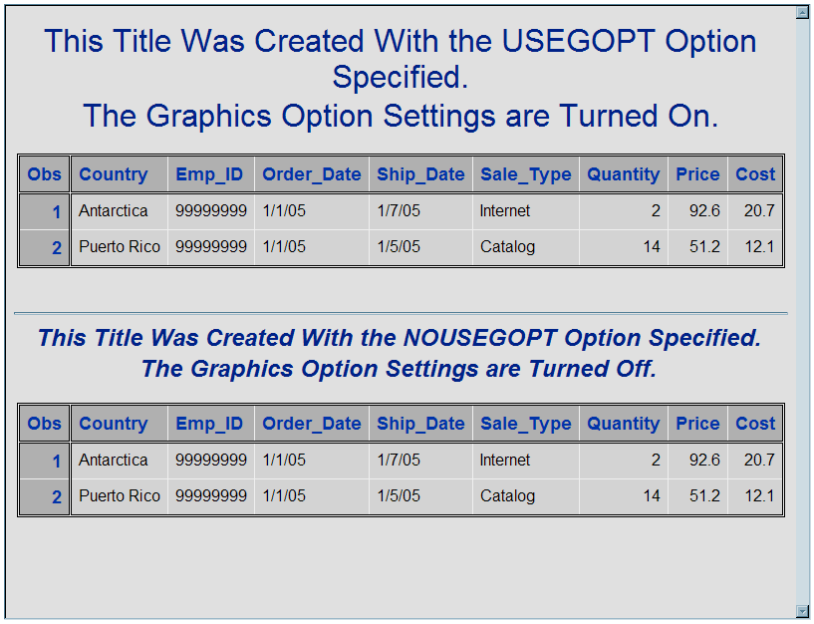
```
title 'This Title Was Created With the NOUSEGOPT Option Specified' ;
title2 'The Graphics Option Settings are Turned Off';
proc print data=exprev (obs=2) ;
run;
```

Close the HTML destination and open the LISTING destination. The ODS HTML CLOSE statement closes the HTML destination. To return ODS to its default setup, the ODS LISTING statement opens the LISTING destination.

```
ods html close;
ods listing;
```

Display 5.28 HTML Output

In the following example, the heights and fonts for the titles of the first table are specified by the FTITLE, FTEXT, and HTEXT options in the GOPTIONS statement. The heights and fonts for the titles of the second table are specified by the default style definition.



This Title Was Created With the USEGOPT Option Specified.
The Graphics Option Settings are Turned On.

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Antarctica	99999999	1/1/05	1/7/05	Internet	2	92.6	20.7
2	Puerto Rico	99999999	1/1/05	1/5/05	Catalog	14	51.2	12.1

This Title Was Created With the NOUSEGOPT Option Specified.
The Graphics Option Settings are Turned Off.

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Antarctica	99999999	1/1/05	1/7/05	Internet	2	92.6	20.7
2	Puerto Rico	99999999	1/1/05	1/5/05	Catalog	14	51.2	12.1

ODS VERIFY Statement

Prints or suppresses a message indicating that a style definition or a table definition being used is not supplied by SAS.

Valid: anywhere

Category: ODS: Output Control

Default: If you do not specify the ODS VERIFY statement, then ODS runs with the verification process turned off. If you specify the ODS VERIFY statement but do not specify an argument, then ODS runs with verification turned on.

See also: For information about how to ignore user-created definitions, see "ODS PATH Statement" on page 206.

Syntax

ODS VERIFY <ON | OFF | ERROR | WARN>;

Options

ON
prints the warning and sends output objects to open destinations.

Alias: ODS VERIFY

Alias: YES

OFF
suppresses the warning.

Alias: ODS NOVERIFY

Alias: NO

ERROR
prints an error message instead of a warning message and does not send output objects to open destinations.

WARN
prints a warning message and does not send output objects to open destinations.

Details

Using the ODS VERIFY Statement PROC TEMPLATE can modify the values in an output object. None of the definitions that SAS provides modifies any values. If you receive a warning from the ODS VERIFY statement, then look at the source code to verify that the values have not been modified.

ODS WML Statement

Opens, manages, or closes the WML destination, which uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a simple list for a table of contents.

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS WML < (<ID=>*identifier*)> *action*;

ODS WML< (<ID=>*identifier*)> <*option(s)*>;

Without an Action or Options

If you use the ODS WML statement without an action or options, then it opens the WML destination and creates WML output.

Actions

The following table lists the actions available for the ODS WML statement. For complete descriptions of actions see “Actions” on page 147 in the ODS MARKUP statement.

Table 5.38 ODS WML Action Summary Table

Task	Action
Close the WML destination and the file that is associated with it	CLOSE
Exclude output objects from the WML destination	EXCLUDE
Select output objects for the WML destination	SELECT
Write to the SAS log the current selection or exclusion list for the WML destination	SHOW

Options

The following table lists the options available for the ODS WML statement, which is part of the markup family of statements. For complete descriptions of these options, see “Options” on page 148 in the ODS MARKUP statement.

Table 5.39 ODS WML Option Summary Table

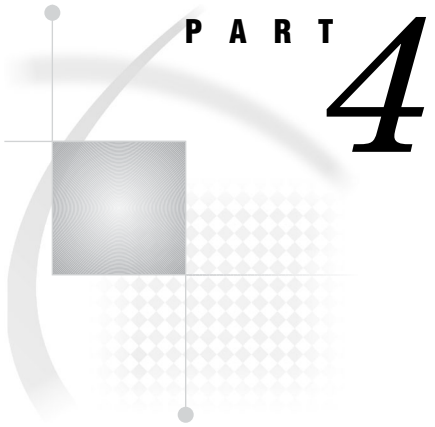
Task	Option
Specify a unique base name for the anchor tag that identifies each output object in the current body file	ANCHOR=
Specify which applet to use to view ODS WML output	ARCHIVE=
Specify attributes to write between the tags that generate dynamic graphics output	ATTRIBUTES=
Specify text to use as the first part of all links and references that ODS creates in output files	BASE=
Open a markup family destination and specify the file that contains the primary output that is created by the ODS statement	BODY=

Task	Option
Specify the character set to be generated in the META declaration for the WML output	CHARSET=
Open the WML destination and specify that the file that contains relevant style information	CODE=
Create a file path that can be used by the GOPTIONS devices	CODEBASE=
Open the WML destination and specify the file that contains a table of contents for the output	CONTENTS=
Specify a cascading style sheet to apply to your output	CSSSTYLE=
Override the encoding for input or output processing (transcodes) of external files	ENCODING=
Specify an event and the value for event variables that is associated with the event	EVENT=
Specify the file that integrates the table of contents, the page contents, and the body file	FRAME=
Control the location where footnotes are printed in the graphics output	GFOOTNOTE | NOGFOOTNOTE
Specify the location for all graphics output that is generated while the destination is open	GPATH=
Control the location where titles are printed in the graphics output	GTITLE | NOGTITLE
Specify WML tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to	HEADTEXT=
Open multiple instances of the same destination at the same time	ID=
Specify WML code to use as the <META> tag between the <HEAD> and </HEAD> tags in all the WML files that the destination writes to	METATEXT=
Create a new body file at the specified starting point. Opens a markup family destination and specifies the file that contains a description of each page of the body file, and contains links to the body file	NEWFILE=
Specify tagset-specific suboptions and a named value	OPTIONS
Specify that the output from the destination be added to an ODS package	PACKAGE
Open the WML destination and specify the file that contains a description of each page of the body file, and contains links to the body file	PAGE=
Write the specified parameters between the tags that generate dynamic graphics output	PARAMETERS=
Specify the location of an aggregate storage location or a SAS catalog for all markup files	PATH=

Task	Option
Specify an alternative character or string to separate lines in the output files	RECORD_SEPARATOR=
Specify a style definition to use in writing output files	STYLE=
Open the WML destination and place style information for output into an external file, or read style sheet information from an existing file	STYLESHEET=
Insert text into your document	TEXT=
Insert into the metadata of a file, a text string that you want to specify as the text to appear in the browser window title bar	TITLE=
Specify a translation table to use when transcoding a file for output	TRANTAB=

Details

The ODS WML statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages, such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.



PART 4

The DOCUMENT Procedure

CHAPTER

6

The DOCUMENT Procedure

Overview: DOCUMENT Procedure

Using the DOCUMENT Procedure

In ODS documents, the DOCUMENT procedure enables you to rearrange, duplicate, or remove output from the results of a procedure or a database query. Also, you can generate output for one or more ODS destinations using the newly transformed output hierarchy file. Thus, the DOCUMENT procedure enables you to do the following:

- transform a report without rerunning an analysis or repeating a database query
- have more control over the structure of output
- display output to any ODS output format without executing SAS programs again
- navigate the current file location and list entries
- open and list ODS documents
- manage output
- store the ODS output objects in raw form

Note: The output is kept in the original internal representation as a data component plus a table template. △

The DOCUMENT procedure is an interactive procedure that enables you to use ODS and global statements within the PROC DOCUMENT step.

Unlike other ODS destinations, the DOCUMENT destination has a graphical user interface (GUI) for performing tasks. However, you can perform the same tasks with batch statement syntax using the DOCUMENT procedure. For a comparison of the

Documents window and the DOCUMENT procedure, see "Comparisons between the Documents Window and the Results Window" on page 371.

DOCUMENT Procedure Terminology

current document	is the open document.
current path	is your current location in the open document. The '^' symbol represents the current path.
entry	is one or more links, output objects, files, or partitioned data sets.
graph segment	is a file type or output object that contains a graph. Graphs are created in some SAS procedures, including those in SAS/GRAPH. The graph output object is referenced as a GRSEG. **See:** For more information about GRSEG and SAS/GRAPH procedures, see *SAS/GRAPH: Reference*.
ODS document	is the hierarchy of output objects that are created by the DOCUMENT procedure. These objects are unformatted and are placed in a SAS item store.
path	is the route through a hierarchal file system, leading to a particular file or file location of an entry within an ODS document. *path* refers to the physical location of an entry. The '^' symbol represents the current path and the '^^' symbol represents the parent path.
replay	is the regeneration of output, in the same or different format, without rerunning analyses or data queries.
root file location	is the top level of a file location in an ODS document. A root file location is not contained within another file location and it does not have a name assigned. A root file location is similar to the root directory of a Windows operating environment.

Syntax: DOCUMENT Procedure

PROC DOCUMENT <*options*>;

COPY *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, …*path-n*<(*where-expression-n*)>> **TO** *path* </ *option(s)*>;

DELETE *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, …*path-n*<(*where-expression-n*)>> < / LEVELS= ALL | *value*>;

DIR <*path*>;

DOC <*options*>;

DOC CLOSE;

HIDE *path* <, *path-2*, …*path-n*>;

IMPORT DATA= *data-set-name* | GRSEG=*grseg* **TO** *path* </*options*>;

LINK *path* TO *path* </ *options*>;

LIST *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>

<, …*path-n*<(*where-expression-n*)>> </*option(s)*>;

MAKE *path <, path-2, …path-n> </ options*>;

MOVE *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, …*path-n*<(*where-expression-n*)>> **TO** *path </ option(s)* >;

NOTE *path <'text'> </ option(s)*>;

OBANOTE<*n*> *output-object <'text'> </option*>;

OBBNOTE<*n*> *output-object <'text'> </ option*>;

OBFOOTN<*n*> *output-object <'text'*>;

OBPAGE *output-object </ option(s)*>;

OBSTITLE<*n*> *output-object <'text'> </ options*>;

OBTEMPL *output-object*;

OBTITLE<*n*> *output-object <'text'*>;

RENAME *path-1* **TO** *path-2*;

REPLAY *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, …*path-n*<(*where-expression-n*)>> </ *options*>;

SETLABEL *path 'label'*;

UNHIDE *path <, path-2, …path-n*>;

QUIT;

Table 6.1 PROC DOCUMENT Statements

Task	Statement
Render ODS output without rerunning procedures and gain more control over the structure and hierarchy of the output	"PROC DOCUMENT Statement" on page 337
Insert a copy of an entry into a specified path	"COPY TO Statement "on page 339
Delete entries from a specified path or paths	"DELETE Statement" on page 340
Set or display the current directory	"DIR Statement" on page 341
Open a document and its contents to browse or edit	"DOC Statement" on page 341
Close the current document	"DOC CLOSE Statement" on page 343
Prevent output from being displayed when the document is replayed	"HIDE Statement" on page 343
Import a data set or graph segment into the current directory	"IMPORT TO Statement" on page 343
Create a symbolic link from one output object to another output object	"LINK Statement" on page 345
List the content of one or more entries	"LIST Statement" on page 346
Create one or more new directories	"MAKE Statement" on page 347
Move entries from one directory to another directory	"MOVE TO Statement" on page 348
Create text strings in the current directory	"NOTE Statement" on page 349
Create or modify lines of text after the specified output object	"OBANOTE Statement" on page 350

Task	Statement
Create or modify lines of text before the specified output object	"OBBNOTE Statement" on page 351
Create or modify lines of text at the bottom of the page in which the output object is displayed	"OBFOOTN Statement" on page 352
Create or delete a page break for an output object	"OBPAGE Statement" on page 353
Create or modify subtitles	"OBSTITLE Statement" on page 354
Write the source code of the ODS template that is associated with a specified output object	"OBTEMPL Statement" on page 355
Create or modify lines of text at the top of the page where the output object is displayed	"OBTITLE Statement" on page 356
Assign a different name to a directory or output object	"RENAME TO Statement" on page 356
Replay one or more entries to the specified open ODS destinations	"REPLAY Statement" on page 357
Assign a label to the current entry	"SETLABEL Statement" on page 358
Enable the output of a hidden entry to be displayed when it is replayed	"UNHIDE Statement" on page 359

PROC DOCUMENT Statement

Creates or opens a document to modify.

Default: Documents are opened in the UPDATE access mode.

Caution: If the DOCUMENT destination is not closed with an ODS DOCUMENT CLOSE statement, then ODS continues to append files to the document.

PROC DOCUMENT <*options* <*access-option(s)*>>;

Without Options

If no options are specified, then the PROC DOCUMENT statement opens the last document that was created in the current SAS session.

Options

NAME= *<libref.>member-name <access-option(s)>*
specifies the name of a new or existing document and its access mode.

<libref.>member-name
identifies a new or existing ODS document.

Default: If no library is specified, then the WORK library is used.

Restriction: The ODS document must be a SAS library member.

access-option(s)
specifies the access mode for the ODS document.
For example, the following PROC DOCUMENT statement opens the document WORK.MyDoc in update mode:

```
proc document name=mydoc;
run;
```

Default: UPDATE

READ
opens a document and provides read-only access.

Requirement: To open a document in the READ access mode, the document must already exist.

Interaction: If a label has been specified with the LABEL= option, then the label is ignored.

WRITE
opens a document and provides Read and Write access.
For example, the following PROC DOCUMENT statement opens the document WORK.YourDoc in Write mode:

```
proc document name=yourdoc(write);
run;
```

Caution: If the ODS document already exists, then it will be overwritten.

Interaction: If a label has been specified with the LABEL= option, then it will override any existing label assigned to the document.

Tip: If the ODS document does not exist, then it will be created.

UPDATE
opens an ODS document and appends new content to the document. UPDATE provides Update access as well as Read access.

Caution: If the document already exists, then its contents will not be changed.

Interaction: If a label has been specified with the LABEL= option, then it will be assigned to the document.

Tip: If the ODS document does not exist, then the document will be created.

LABEL= *'label'*
assigns a label to a document.
For example, the following PROC DOCUMENT statement opens the document WORK.YourDoc in Write mode and assigns a label to it:

```
proc document name=yourdoc(write) label='repeated measures results';
run;
```

Restriction: Labels can be assigned only to documents with Write-access permissions.

Requirement: Enclose labels in quotation marks.

COPY TO Statement

Copies an entry into the specified path.

Default: If you do not specify a location to insert the entry into the path, then the entry is inserted at the end of the path.

COPY *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, *...path-n*<(*where-expression-n*)>> **TO** *path* </*option(s)*>;

Required Arguments

path
is the location where a link, output object, or file is copied.

Requirement: Separate multiple paths with commas.

Tip: The '^' symbol represents the current path and the '^^' symbol represents the parent path.

Options

AFTER= *path*
inserts a copy of an entry after the specified path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

BEFORE= *path*
inserts a copy of an entry before the specified path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

FIRST
inserts a copy of an entry at the beginning of the specified path.
For example, the following COPY TO statement inserts a copy of the entry Monday_Report at the beginning of the root directory:

```
copy weekly\monday_report to \ /first;
run;
```

LAST
inserts a copy of an entry at the end of the specified path.

LEVELS= ALL | *value*
specifies the number of levels below the specified path that you want to copy.

ALL
specifies all levels of the path.

value
specifies the numeric value of the path level.

For example, the following COPY TO statement copies two levels of the entry Weekly to the entry Monthly:

```
copy weekly to \work.mydoc\monthly /levels = 2;
run;
```

Default: ALL

Restriction: The LEVELS= option is valid only when you specify a directory.

where-expression
selects, for copying, entries in an ODS document that meet a particular condition.

See: "Using WHERE Expressions with the DOCUMENT Procedure" on page 361

DELETE Statement

Deletes entries from the current file location.

Restriction: The root file location cannot be deleted or moved.

Caution: The DELETE statement affects all levels of a file location below the specified path.

DELETE *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, …*path-n*<(*where-expression-n*)>> </ LEVELS= ALL | *value*>;

Required Arguments

path
specifies the location of one or more links, output objects, or file locations.
For example, the following DELETE statement removes the ClassLevels and Nobs entries from the current directory:

```
delete classlevels, nobs;
run;
```

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

LEVELS= ALL | *value*
specifies the level of the path that you want to delete.

ALL
specifies all levels of the path.

value
specifies the numeric value of the path level.

Default: ALL

Restriction: The LEVELS= option is valid only when you specify a directory.

where-expression
selects, for deletion, entries in an ODS document that meet a particular condition.

See also: "Using WHERE Expressions with the DOCUMENT Procedure" on page 361

DIR Statement

Sets or displays the current file location.

Featured in: Example 1 on page 374, Example 2 on page 378, and Example 3 on page 381

DIR <*path*>;

Without Options

If no options are specified, then the DIR statement displays the current path.

Options

path
sets the current file location.
For example, the following DIR statement sets the current directory to '\report\glm' within the current document:

```
dir \report\glm;
run;
```

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

DOC Statement

Opens a document and its contents to browse or edit.

Default: Documents are opened in the UPDATE access mode.

Featured in: Example 1 on page 374 and Example 2 on page 378

DOC <*options* <*access-option(s)*>>;

Without Options

If no options are specified, then the DOC statement lists the ODS documents in all SAS libraries in alphabetical order. Document labels, if any, are displayed.

Options

LABEL= *'label'*
assigns a label to a document.
For example, the following DOC statement opens the document WORK.YourDoc in Write mode and assigns a label to it:

```
doc name=yourdoc(write) label='repeated measures results';
run;
```

Restriction: A label can be assigned only to documents with Write access permission.

Requirement: To use the LABEL= option, specify the NAME= option on the DOC statement.

Requirement: Enclose labels in quotation marks.

LIBRARY=*library-name*
specifies that only the documents in the specified *library-name* are listed.

Alias: LIB=

Interaction: The LIBRARY= option cannot be specified with the NAME= or LABEL= options.

NAME= *libref.member-name <access-option(s)>*
specifies the name that you assign to a document and its access mode.

<libref.>member-name
identifies a document.

Default: If no library is specified, then the WORK library is used.

Restriction: The document must be a SAS library member.

access-option(s)
specifies the access mode for the document.

READ
opens a document and provides read-only access.

Interaction: If a label has been specified with the LABEL= option, then the label is ignored.

WRITE
opens a document and provides Write access, but only if you have Write permission.

Caution: If the document already exists, then it will be overwritten. If the document does not exist, then it will be created.

Interaction: If a label has been specified with the LABEL= option, then it will override any existing label assigned to the document.

UPDATE
opens a document and provides Update access, but only if you have Update permission.

Interaction: If a label has been specified with the LABEL= option, then it will be assigned to the document.

Tip: If the document already exists, then its contents will not be changed and the new contents will be appended to the document. If the document does not exist, then it will be created.

DOC CLOSE Statement

Closes the current document.

DOC CLOSE;

HIDE Statement

Prevents output from being displayed when the document is replayed.

Tip: To see entries that might be hidden in the current document, use the LIST statement.

HIDE *path* <, *path-2*, …*path-n*>;

Required Arguments

path
specifies the location of the file or files that you want to hide.

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

IMPORT TO Statement

Imports the specified SAS data set or graph segment to the current file location.

IMPORT DATA= *data-set-name*<*data-set-option(s)*>| GRSEG=*grseg* **TO** *path* </ *option(s)*>;

Required Arguments

DATA= *data-set-name*
specifies an existing SAS data set that you want to import.

GRSEG= *grseg*
stores a reference to a graph segment.

grseg
specifies the 3-level catalog path name. For example, GRSEG=SASUSER.grseg.mygraph.

See: GRSEG= option in the *SAS/GRAPH: Reference*

path
specifies the location where you want to import the data set or graph segment.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

AFTER= *path*
imports the data set or graph segment into the file location after the specified path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

BEFORE= *path*
imports the data set or graph segment into the file location before the specified path. For example, the following IMPORT TO statement imports the data set SASHELP.Class to the current directory, and inserts the data set before the entry MyInfo:

```
import data=sashelp.class to ^ /before=MyInfo;
run;
```

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

data-set-option(s)
specify actions that apply only to the SAS data set.

See also: *SAS Language Reference: Dictionary* for information about SAS data sets and their options

FIRST
imports the data set or graph segment at the beginning of the file location.

LAST
imports the data set or graph segment at the end the file location.

LINK Statement

Creates a symbolic link from one specified output object to another in the current file location.

LINK *path* TO *path* </ *option(s)*>;

Required Arguments

path
specifies the locations of the output objects that you want to link to one another.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

AFTER= *path*
links to the entry that follows the specified path in the current file location.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

BEFORE= *path*
links to the entry that precedes the specified path in the current file location.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

FIRST
links to the first entry in the current file location.

HARD
specifies a type of link that refers to a copy of an output object within the ODS document. All data is shared between the link and the target, except names and labels.
For example, the following LINK statement creates a hard link from the output object ErrorSSCP to the output object LinkedErrorSSCP in the current directory:

```
link errorSSCP to linkederrorSSCP /hard;
run;
```

Restriction: A hard link can reference only an output object, and the source and target paths must be in the same ODS document. The target must exist when you create the hard link.

Interaction: A hard link and its target exist independently. Deleting a hard link does not affect the target. Similarly, deleting a target does not affect the link.

LABEL
copies the source label to the link.

Default: The source label is not copied unless the LABEL option is specified.

LAST
links to the last entry in the current file location.

LIST Statement

Lists the contents of one or more entries.

Default: Only summary information is displayed if the DETAILS option is omitted.

Default: If the ORDER= option is omitted, then the contents of the specified entries are listed in the order specified by the INSERT option.

Tip: To see any entries that might be hidden in the current file location, use the LIST statement.

Featured in: Example 1 on page 374, Example 2 on page 378, and Example 3 on page 381

LIST *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, ...*path-n*<(*where-expression-n*)>> </ *option(s)*>;

Required Arguments

path
specifies the location of an entry. An entry can be one or more file locations, links, or output objects.
For example, the following LIST statement lists all of the entries within the Report entry:

```
list \sasuser.imports\report;
run;
```

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

BYGROUPS
creates, in the entry list, columns for BY variables. The name of the BY variable becomes the column name. The values of the BY variables are listed in the columns.

Note: When you specify the BYGROUPS option, only entries containing BY group information are listed. △

Interaction: It is recommended that when you specify the BYGROUPS option, specify the LEVELS=ALL option also. If the LEVELS=ALL option is not specified, then ODS cannot find BY group information within all levels of the directories. Therefore, no list can be produced because only entries containing BY group information are listed.

Featured in: Example 4 on page 387

DETAILS
specifies the properties of the entries.
For example, the following LIST statement lists the details of three levels of the Report entry:

```
list \sasuser.imports\report /details levels=3;
run;
```

FOLLOW
resolves all links and lists the contents of the entries.

LEVELS= ALL | *value*
specifies the level of the path that you want to list.

ALL
specifies all levels of the path.

value
specifies the numeric value of the path level.
For example, the following LIST statement lists the details of three levels of the Report entry:

```
list \sasuser.imports\report /details levels=3;
run;
```

Default: If you omit the LEVELS= option, then the default value of the level is 1.

Restriction: The LEVELS= option is valid only when you specify a directory.

ORDER= ALPHA | DATE | INSERT
specifies the order in which the entries are listed.

ALPHA
lists the entries in alphabetical order.

DATE
lists the file locations in ascending order based on the date and time the files were created.

INSERT
lists the file locations in the order in which you arranged the entries.

where-expression
selects, for listing, entries in an ODS document that meet a particular condition.

Featured in: Example 2 on page 378

See: "Using WHERE Expressions with the DOCUMENT Procedure" on page 361

MAKE Statement

Creates one or more new file locations.

Default: If no location is specified, the newly created file location is appended to the end of the current file location.

MAKE *path <, path-2, …path-n> </ option(s)>*;

Required Arguments

path
specifies the newly created file location.

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

AFTER= *path*
adds the newly created file location after the specified path in the current file location.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

BEFORE= *path*
adds the newly created file location before the specified path in the current file location.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

FIRST
adds the newly created file location to the beginning of the current file location.

LAST
adds the newly created file location to the end of the current file location.

MOVE TO Statement

Moves entries from the specified location to another location.

Restriction: The root file location cannot be moved or deleted.

Requirement: Separate multiple paths with commas.

Caution: The MOVE TO statement affects all levels of a file location below the specified starting level.

MOVE *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, …*path-n*<(*where-expression-n*)>> **TO** *path* </ *option(s)*>;

Required Arguments

path
specifies the location of links, output objects, or files that you want to move.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

AFTER= *path*
moves the entry after the specified entry in the path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

BEFORE= *path*
moves the entry before the specified entry in the path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

FIRST
moves the entry to the beginning of the specified file location.

LAST
moves the entry to the end of the specified file location.

LEVELS= ALL | *value*
specifies the level of the path that you want to move.

ALL
specifies all levels of the path.

value
specifies the numeric value of the path level.
For example, the following MOVE TO statement moves two levels of the directory Weekly to the Monthly directory of WORK.MyDoc:

```
move weekly to \work.mydoc\monthly /levels = 2;
run;
```

Default: ALL

Restriction: The LEVELS= option is valid only when you specify a directory.

where-expression
selects, for moving, entries in an ODS document that meet a particular condition.

See: "Using WHERE Expressions with the DOCUMENT Procedure" on page 361

NOTE Statement

Creates text strings in the current file location.

Default: If you omit the JUST= option, then the note is centered between the left and right margins.

Default: If no location is specified, then the note is added to the end of the current location.

Featured in: Example 3 on page 381

NOTE *path <'text'> </ option(s)>*;

Without Options

If no text string is specified, then the NOTE statement creates a blank note.

Required Arguments

path
specifies the location where the note is stored.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Options

AFTER= *path*
inserts the text string after the specified path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

BEFORE= *path*
inserts the text string before the specified path.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

FIRST
inserts the text string at the beginning of the path.

JUST= LEFT | CENTER | RIGHT
specifies the alignment of the text string.

LEFT
aligns the text string with the left margin.

CENTER
centers the text string between the left and right margins.

RIGHT
aligns the text string with the right margin.

LAST
inserts the text string at the end of the path.

'text'
specifies the text string.

Requirement: All text strings must be enclosed in quotation marks.

OBANOTE Statement

Creates or modifies an object footer (lines of text) after the specified output object.

Featured in: Example 3 on page 381

OBANOTE<*n*> *output-object* <*'text'*> </ JUST= LEFT | CENTER | RIGHT>;

Required Arguments

output-object
specifies the name of the ODS output object.

Options

JUST= LEFT | CENTER | RIGHT
specifies the alignment of the object footer.

LEFT
aligns the object footer with the left margin.

CENTER
centers the object footer between the left and right margins.

RIGHT
aligns the object footer with the right margin.

n
specifies the relative line that contains the object footer.

Default: If you omit *n*, then SAS assumes a value of 1. Therefore, specify either OBANOTE or OBANOTE1 for the first text line.

Range: 1–10

Tip: The OBANOTE line with the highest number appears on the bottom line.

Tip: You can create notes that contain blank lines between them. For example, if you specify a text string with an OBANOTE1 statement that is followed by an OBANOTE3 statement, then a blank line separates the two lines of text.

'text'
specifies the text string that becomes the object footer.

You can customize object footers by inserting BY variable values (#BYVALn), BY variable names (#BYVARn), or BY lines (#BYLINE) into object footers that are specified in PROC DOCUMENT steps. After you specify the object footer, embed the items at the position where you want them to appear. For more information, see "Customizing Labels, Titles, and Footnotes with BY Variables" on page 359.

Requirement: All text strings must be enclosed in quotation marks.

Caution: If no text string is specified, then the OBANOTE statement deletes all object footers for the specified output object only.

OBBNOTE Statement

Creates or modifies an object heading (lines of text) before the output object.

Featured in: Example 3 on page 381

OBBNOTE<*n*> *output-object* <*'text'*> </ JUST= LEFT | CENTER | RIGHT>;

Required Arguments

output-object
specifies the name of the ODS output object.

Options

JUST= LEFT | CENTER | RIGHT
specifies the alignment of the object heading.

LEFT
aligns the object heading with the left margin.

CENTER
centers the object heading between the left and right margins.

RIGHT
aligns the object heading with the right margin.

n
specifies the relative line that contains the object heading.

Default: If you omit *n*, then SAS assumes a value of 1. Therefore, specify either OBBNOTE or OBBNOTE1 for the first text line.

Range: 1– 10

Tip: The OBBNOTE line with the highest number appears on the bottom line.

Tip: You can create notes that contain blank lines between them. For example, if you specify a text string with an OBBNOTE statement that is followed by an OBBNOTE3 statement, then a blank line separates the two lines of text.

'text'
specifies the text string that becomes the object heading.

You can customize object headings by inserting BY variable values (#BYVALn), BY variable names (#BYVARn), or BY lines (#BYLINE) into object headings that are specified in PROC DOCUMENT steps. After you specify the object heading text, embed the items at the position where you want them to appear. For more information, see "Customizing Labels, Titles, and Footnotes with BY Variables" on page 359.

Requirement: All text strings must be enclosed in quotation marks.

Caution: If no text string is specified, then the OBBNOTE statement deletes all existing object headings for the specified output object only.

OBFOOTN Statement

Creates or modifies lines of text at the bottom of the page on which the output object is displayed.

Restriction: You can print up to ten lines of text.

Tip: The OBFOOTN statement is similar to the global FOOTNOTE statement.

Featured in: Example 3 on page 381

OBFOOTN<*n*> *output-object* <*'text'*>;

Required Arguments

output-object
specifies the ODS output object.

Options

n
specifies the relative line that contains the footnote.

Range: 1–10

Tip: The OBFOOTN line with the highest number appears on the bottom line. If you omit *n*, then SAS assumes a value of 1. Therefore, specify OBFOOTN or OBFOOTN1 for the first text line.

Tip: You can create footnotes that contain blank lines between them. For example, if you specify a text string with an OBFOOTN statement that is followed by an OBFOOTN3 statement, then a blank line separates the two lines of text.

'text'
specifies the text string that becomes the footnote.

You can customize footnotes by inserting BY variable values (#BYVALn), BY variable names (#BYVARn), or BY lines (#BYLINE) into footnotes that are specified in PROC DOCUMENT steps. After you specify the text, embed the items at the position where you want them to appear. For more information, see "Customizing Labels, Titles, and Footnotes with BY Variables" on page 359.

Requirement: All text strings must be enclosed by quotation marks.

Caution: If you use the OBFOOTN statement without a text string, then all existing footnotes for the specified output object are deleted.

OBPAGE Statement

Creates or deletes a page break for an output object.

Featured in: Example 3 on page 381

OBPAGE *output-object*< / <DELETE > <AFTER>>;

Required Arguments

output-object
specifies the name of the output object.

Without Options

If no options are specified, then the OBPAGE statement inserts a page break before an output object.

Options

AFTER
inserts a page break after an output object.

Tip: To delete a page break after an output object, use the AFTER option as well as the DELETE option.

DELETE
removes the page break for an output object.

OBSTITLE Statement

C

Featured in: Example 3 on page 381

OBSTITLE<*n*> *output-object* <*'text'*> </ JUST= LEFT | CENTER | RIGHT>;

Required Arguments

output-object
specifies the ODS output object.

Options

JUST= LEFT | CENTER | RIGHT
specifies the alignment of the text string.

LEFT
aligns the text string with the left margin.

CENTER
aligns the text string in the center between the left and right margins.

RIGHT

aligns the text string with the right margin.

n
specifies the relative line that contains the subtitle.

Range: 1–10

Tip: The OBSTITLE line with the highest number appears on the bottom line. If you omit *n*, then SAS assumes a value of 1. Therefore, you can specify OBSTITLE or OBSTITLE1 for the first text line.

Tip: You can create subtitles that contain blank lines between them. For example, if you specify a text string with an OBSTITLE statement that is followed by an OBSTITLE3 statement, then a blank line separates the two lines of text.

'text'
specifies the text string.

You can customize subtitles by inserting BY variable values (#BYVALn), BY variable names (#BYVARn), or BY lines (#BYLINE) into subtitles that are specified in PROC DOCUMENT steps. After you specify text, embed the items at the position where you want them to appear. For more information, see "Customizing Labels, Titles, and Footnotes with BY Variables" on page 359.

Requirement: All text strings must be enclosed in quotation marks.

Caution: If no arguments are specified, then the OBSTITLE statement deletes all existing subtitles for the specified output object only.

OBTEMPL Statement

Writes, to any open ODS destination, the source code of the ODS template that is associated with the specified output object.

Restriction: If the output object that is specified has no ODS template associated with it, then no output is created.

OBTEMPL *output-object*;

Required Arguments

output-object
specifies the path name of the output object.

See also: "Getting Familiar with Output Objects" on page 366

Featured in: Example 4 on page 387

OBTITLE Statement

Creates or modifies title lines for the output.

Tip: The OBTITLE is similar to the global TITLE statement.

Featured in: Example 3 on page 381

OBTITLE<*n*> *output-object* <*'text'*>;

Required Arguments

output-object
specifies the name of the output object.

Options

n
specifies the relative line that contains the title.

Range: 1–10

Tip: The OBTITLE line with the highest number appears on the bottom line. If you omit *n*, then SAS assumes a value of 1. Therefore, specify OBTITLE or OBTITLE1 for the first text line.

Tip: You can create titles that contain blank lines between them. For example, if you specify a text string with an OBTITLE statement that is followed by an OBTITLE3 statement, then a blank line separates the two lines of text.

'text'
specifies the text string.

You can customize titles by inserting BY variable values (#BYVALn), BY variable names (#BYVARn), or BY lines (#BYLINE) into output titles that are specified in PROC DOCUMENT steps. After you specify the text, embed the items at the position where you want them to appear. For more information, see "Customizing Labels, Titles, and Footnotes with BY Variables" on page 359.

Requirement: All text strings must be enclosed in quotation marks.

Caution: If no text is specified, then the OBTITLE statement deletes all existing titles for the specified output object only.

RENAME TO Statement

Assigns a different name to a file location or output object.

RENAME *path-1* **TO** *path-2*;

Required Arguments

path-1
specifies the current file location or output object.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

path-2
specifies the new name of the file location or output object.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

REPLAY Statement

Displays one or more entries to the specified open ODS destination(s).

Default: If you omit the LEVELS= option, then all levels of the file are displayed to all open destinations.

Featured in: Example 2 on page 378 and Example 3 on page 381

REPLAY *path*<(*where-expression*)> <, *path-2*<(*where-expression-2*)>>
<, *...path-n*<(*where-expression-n*)>> </ *option(s)*>;

Options

ACTIVEFOOTN
specifies that footnotes that are active in a SAS session will override the footnotes that are stored in an ODS document.

Alias: ACFOOTN

ACTIVETITLE
specifies that titles that are active in a SAS session will override the titles that are stored in an ODS document.

Alias: ACTITLE

DEST= (*ODS-destination(s)*)
specifies one or more ODS destinations to display the output objects.
For example, the following REPLAY statement replays two levels of the entry Data to the HTML and RTF destinations:

```
replay \Report\GLM#1\Data /levels=2 dest=(html rtf);
run;
```

Requirement: When you specify the DEST= option, surround the ODS destinations with parentheses and separate each destination with a blank space. For example, DEST=(HTML RTF LISTING)

Tip: When you specify only one destination, you do not need to use parentheses. For example, DEST=HTML

See also: For information about ODS destinations, see "Understanding ODS Destinations" on page 24.

LEVELS= ALL | *value*
specifies the level of the path that you want to replay.

ALL
specifies that all levels of the path are displayed to all open destinations.

value
specifies the numeric value of the path level.
For example, the following REPLAY statement replays two levels of the entry Data to the HTML and RTF destinations:

```
replay \Report\GLM#1\Data /levels=2 dest=(html rtf);
run;
```

Default: ALL

Restriction: The LEVELS= option is valid only when you specify a directory.

path
specifies the location of an entry. An entry can be one or more file locations, links, or output objects.

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

where-expression
selects, for replaying, entries in an ODS document that meet a particular condition.

See: "Using WHERE Expressions with the DOCUMENT Procedure" on page 361

Replaying Graphics

When replaying graphics created by a device driver from the following list, you must also specify a device driver from the list with the DEVICE= option in the GOPTIONS statement:

- ACTIVEX
- ACTXIMG
- JAVA
- JAVAIMG

See the GOPTIONS statement in *SAS/GRAPH: Reference* for more information.

SETLABEL Statement

Assigns a label to the specified path.

SETLABEL *path 'label'*;

Required Arguments

'label'
specifies the text of the label. You can customize labels by inserting BY variable values (#BYVAL), BY variable names (#BYVAR), or BY lines (#BYLINE) into labels that are specified in PROC DOCUMENT steps.

Requirement: The label must be enclosed in quotation marks.

See also: For more information, see "Customizing Labels, Titles, and Footnotes with BY Variables" on page 359.

path
specifies the location of a link, output object, or file location.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

UNHIDE Statement

Enables the output of a hidden entry to be displayed when it is replayed.

UNHIDE *path <, path-2, …path-n>*;

Required Arguments

path
specifies the location of a link, output object, or file.

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol '^' to represent the current path and the symbol '^^' to represent the parent path.

Customizing Labels, Titles, and Footnotes with BY Variables

You can customize labels, titles, and footnotes with these statements by inserting BY variable values (#BYVAL), BY variable names (#BYVAR), or BY lines (#BYLINE) in labels that are specified in the following PROC DOCUMENT statements:

"OBANOTE Statement" on page 350
"OBBNOTE Statement" on page 351
"OBFOOTN Statement" on page 352
"OBSTITLE Statement" on page 354
"OBTITLE Statement" on page 356
"SETLABEL Statement" on page 358

Note: The #BYVAL, #BYVAR, and #BYLINE substitutions only show up for replayed output objects that belong to a BY group. Examples of output objects that do not belong to a BY group are data sets that are imported into a document with the IMPORT TO statement, and notes that are created with the NOTES statement. △

To create these substitutions, embed the items in the specified object text string at the position where you want the substitution text to appear. The #BYVAL, #BYVAR, and #BYLINE substitutions have this form:

#BYVAL*n* | #BYVAL(*variable-name*)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value in the label.

Follow these rules when you use #BYVAL in a statement of a PROC DOCUMENT step:

- Specify the variable that is used by #BYVAL in the BY statement.
- Insert #BYVAL in the specified text string at the position where you want the substitution text to appear.
- Follow #BYVAL with a delimiting character, either a space or other nonalphanumeric character (for example, a quotation mark) that ends the text string.
- To immediately follow the #BYVAL substitution with other text and no delimiter, use a trailing dot (as with macro variables).
- Specify the variable with one of the following:

  n
  specifies which variable in the BY statement that #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement.

  Example: #BYVAL2 specifies the second variable in the BY statement.

  variable-name
  names the BY variable.

  Example: #BYVAL(YEAR) specifies the BY variable, YEAR.

  Tip: *variable-name* is not case sensitive.

  Requirement: You must enclose *variable-name* in parentheses.

#BYVAR*n* | #BYVAR(*variable-name*)
substitutes the name of the BY variable or label that is associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label.

Follow these rules when you use #BYVAR in a statement of a PROC DOCUMENT step:

- Specify the variable that is used by #BYVAR in the BY statement.
- Insert #BYVAR in the specified text string at the position where you want the substitution text to appear.
- Follow #BYVAR with a delimiting character, either a space or other nonalphanumeric character (for example, a quotation mark) that ends the text string.
- To immediately follow the #BYVAR substitution with other text and no delimiter, use a trailing dot (as with macro variables).

□ Specify the variable with one of the following:

n
specifies the variable in the BY statement that #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement.

Example: #BYVAR2 specifies the second variable in the BY statement.

variable-name
names the BY variable.

Example: #BYVAR(SITES) specifies the BY variable SITES.

Tip: *variable-name* is not case sensitive.

Requirement: You must enclose *variable-name* in parentheses.

#BYLINE
substitutes the entire BY line without leading or trailing blanks for #BYLINE in the text string and displays the BY line in the label.

Using WHERE Expressions with the DOCUMENT Procedure

You can conditionally select a subset of entries in an ODS document for copying, listing, deleting, moving, or replaying by using WHERE expressions with the following statements:

"COPY TO Statement "on page 339

"DELETE Statement" on page 340

"LIST Statement" on page 346

"MOVE TO Statement" on page 348

"REPLAY Statement" on page 357

WHERE expressions have this form:

(**WHERE**=(*where-expression-1 <operator where-expression-n>*))

where-expression
is an arithmetic or logical expression that consists of a sequence of operators and operands.

operand
is one of the following:

constant
is a fixed value such as a date literal, a value, or a BY variable.

SAS function
For information on SAS functions, see *SAS Language Reference: Dictionary*.

subsetting variable
is a special kind of WHERE expression operand used by the DOCUMENT procedure to help you find common values in ODS documents. Here are the subsetting variables:

CDATE
is the creation date of the current entry.

Example: The following MOVE TO statement moves all entries of the type 'Graph' with a creation date of 16JUL2004 to the Monthly directory of WORK.MyDoc:

```
move ^(where=(_type_ = 'Graph' and _cdate_ = '16JUL2004'd)) to
   \ work.mydoc\monthly;
run;
```

CDATETIME
is the creation datetime of the current entry.

Example: The following COPY TO statement copies all entries with a creation datetime of May 1, 2003, at 9:30 to the Monthly directory of WORK.MyDoc:

```
copy ^(where=(_cdatetime_ = '01may04:9:30:00'dt)) to \work.mydoc\monthly;
run;
```

CTIME
is the creation time of the current entry.

Example: The following DELETE statement deletes all entries with a creation time of 9:25:19 PM:

```
delete ^(where=(_ctime_ = '9:25:19pm't));
run;
```

LABEL
is the label of the current entry.

Example: The following LIST statement lists all tables containing the label 'Type III Model' within the GLM procedure:

```
list glm(where=(_type_ = 'table' _label_ ? 'Type III Model'));
run;
```

LABELPATH
is the path to the label of the current entry. Document label paths are formed by concatenating the labels and sequence numbers, and then separating them with the forward slash (**/**) symbol. Document label paths are similar to the label paths specified by the ODS TRACE statement.

For example, if the ODS TRACE label path is:

```
'The Univariate Procedure'.'Normal_x'.'Histogram 1'
```

The corresponding document label path is:

```
'The Univariate Procedure'#1\'Normal_x'#1\'Histogram 1'#1
```

Note that in document label paths, the instances of '**.**' are replaced with '****'.

Example: The following LIST statement lists all items containing "Fit Statistics" in the label path.

```
list gml(where=(_labelpath_ ? "Fit Statistics"))/levels=all;
run;
```

See also: "ODS TRACE Statement" on page 317

MDATE
is the modification date of the current entry.

Example: The following MOVE TO statement moves all entries of the type 'Graph' with a modification date of 16JUL2004 to the Monthly directory of WORK.MyDoc:

```
move ^(where=(_type_ = 'Graph' and _mdate_ = '16JUL2004'd)) to
   \work.mydoc\monthly;
run;
```

MDATETIME
is the modification datetime of the current entry.

Example: The following REPLAY statement replays all entries with a modification datetime of May 1, 2003, at 9:30:

```
replay ^(where=(_mdatetime_ = '01may04:9:30:00'dt));
run;
```

MTIME
is the modification time of the current entry.

Example: The following COPY TO statement copies all entries with a modification time of 9:25:19 PM to the Monthly directory of WORK.MyDoc:

```
copy ^(where=(_mtime_ = '9:25:19pm't)) to \work.mydoc\monthly;
run;
```

NAME
is the name of the current entry.

Example: The following DELETE statement deletes all entries that contain the name "stemleng" within the GLM procedure:

```
delete glm(where=(_name_ ? 'stemleng'));
```

PATH
is the path of the current entry.

Example: The following LIST statement lists all entries with a path containing the substring 'Anova' at all levels of the current directory:

```
list ^(where=(_path_ ? 'Anova'));
run;
```

SEQNO
is the sequence number of the current entry.

Example: The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:

```
replay glm(where=(_seqno_ = 2));
```

See also: "Understanding Sequence Numbers" on page 366

TYPE
is the type of the current entry.

Example: The following MOVE TO statement moves all entries of the type 'Graph' with a creation date of July 16, 2004, to the Monthly directory of WORK.MyDoc:

```
move ^(where=(_type_ = 'Graph' and _cdate_ = '16JUL2004'd)) to
     \work.mydoc\monthly;
run;
```

variable- name
is the name of a BY variable.

Example: The following MOVE TO statement moves all entries where the value of the variable Gender is 'F' to the Monthly directory of WORK.MyDoc:

```
move ^(where=(gender='F')) to \work.mydoc\monthly;
run;
```

operator
compares one variable with a value or another variable. *operator* can be AND, OR NOT, OR, AND NOT, or a comparison operator.

Table 6.2 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Requirement: Enclose *where-expression* in quotation marks.

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE= data set option, see the WHERE= data set option in *SAS Language Reference: Dictionary* and "WHERE-Expression Processing" in *SAS Language Reference: Concepts*.

Concepts: DOCUMENT Procedure

About ODS Documents

Definition

An ODS document is a hierarchical file of output objects that is created from a procedure or data query. The output objects are in unformatted form, and they are stored in a SAS item store. The hierarchy is controlled by the internal logic of the procedure or data query.

Items Included in an ODS Document

In an ODS document, each level of the hierarchical file represents a path which refers to the location of a file, link, or output object. An output object can be one of the following:

- table
- graph
- equation
- note

Items Not Included in an ODS Document

An ODS document does not store the following items:

- SAS logs
- SAS system options
- procedure options
- ODS options
- SAS/GRAPH options
- SAS/GRAPH external graph titles
- GRSEGs (references to GRSEGs, but not GRSEGs themselves, are stored)

ODS Document Persistence

An ODS document is a member of a SAS library. Therefore, you can browse, edit, and replay the output contained in the ODS document to any ODS destination without rerunning the SAS programs that created the initial output. An ODS document persists in the SAS System until the document, or the SAS library containing the document, is deleted. Thus an ODS document that was created in the SASUSER library, or in another permanent SAS library, can persist indefinitely because it is considered a permanent archive of SAS procedure output. However, an ODS document that is created in the WORK library does not persist longer than the SAS session that created it. For information about SAS libraries, see *SAS Language Reference: Concepts*.

Understanding an ODS Document Path

Definition of ODS Document Path

Because an ODS document is stored as an item store, this file format enables client applications to define a "hierarchal file system within a file." This is similar to a directory system in a Windows operating environment, or a partitioned data set in a mainframe operating environment. Therefore, an ODS document path means the location of an entry.

Entry Names

Entry names follow these rules:

- must be alphanumeric
- must begin with an alphabetical character
- can contain underscores
- can have no more than 32 characters
- are preserved with casing (uppercase, lowercase, or mixed case) that is specified in the operating environment
- can have labels which are no more than 256 characters

Entries are inserted into an ODS document in the following three ways:

- ordered by insertion, which is the default order
- ordered by ascending date-and-time stamp
- ordered alphabetically

Understanding Sequence Numbers

Entry names are not required to be unique within an ODS document. However, they are uniquely identifiable because they contain sequence numbers. Every entry in an ODS document, except for the root file location, has a sequence number. A sequence number is a positive integer that is unique with respect to the name of the entry within the same file location level. Entries are assigned sequence numbers according to the sequence in which they are added to a file location. For example, the first entry **myname** is assigned a sequence number 1, **myname#1**. The second entry **myname** is assigned a sequence number 2, **myname#2**. Sequence numbers are never reassigned, unless all entries with the same name are deleted. In this case, the sequence numbers are reset to an initial number of 1.

ODS Documents and Base SAS Procedures

You can create an ODS document from almost any Base SAS procedure. The PRINT, REPORT, and TABULATE procedures use table templates that are created by the user, and not defined by a template in ODS. These procedures use custom table templates, custom data components, and custom formats for their output objects. Nevertheless, the ODS document and all of its features are supported for the TABULATE and REPORT procedures. ODS documents do not support some features of PROC PRINT. For example, BY group processing in the PRINT procedure is not supported.

Getting Familiar with Output Objects

An output object is one of the following:

- equation
- graph
- note
- table

Output objects have associated information and attributes. Some or all of these attributes pertain to output objects.

after-note is the note assigned to the output object by the procedure that produced the object. This note is displayed every time the output object is displayed. After-notes display after the output object.

before-note is the note assigned to the output object by the procedure that produced the object. This note is displayed every time the output object is displayed. Before-notes display before the output object.

footnote is created by the FOOTNOTE statement and is displayed when the output object is created.

page break causes a page break prior to displaying the output object and any associated titles and notes.

subtitle is the title that is assigned to the output object by the procedure that produced the output object. This title is displayed every time a new page of output is started.

title is created by the TITLE statement and is displayed when the output object is created.

Here is the order in which the attributes of an output object are displayed:

1. page break
2. titles
3. subtitles
4. before-notes
5. output object
6. after-notes
7. footnotes

Understanding How ODS Documents Interact across Operating Environments

Compatibility across SAS Versions

An ODS document that is created in the current version of SAS is compatible with later versions of SAS. In most cases, an ODS document created in a later version of SAS will still be compatible with an earlier version of SAS.

ODS documents are not portable across operating environments. For example, an ODS document created in a Windows operating environment cannot be used in a mainframe operating environment.

Results: DOCUMENT Procedure

ODS Documents in the Documents Window

Understanding When to Use the Documents Window

The Documents window displays ODS documents in a hierarchical tree structure. The Documents window does the following:

- displays all ODS documents, including ODS documents stored in SAS libraries
- organizes, manages, and customizes the layout of the entries contained in ODS documents
- displays the property information of ODS documents
- replays entries
- renames, copies, moves, or deletes ODS documents
- creates shortcuts to ODS documents

For a comparison of the Documents window to the Results Window, see “Comparisons between the Documents Window and the Results Window” on page 371.

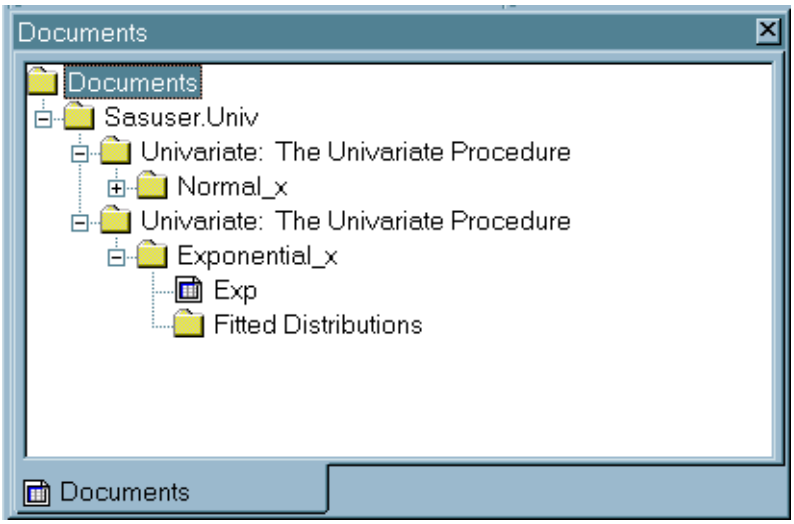
Viewing an ODS Document in the Documents Window

To view the Documents window, submit this command in the command bar:

```
odsdocuments
```

This display shows the Documents window that contains the ODS document named **Sasuser.Univ**. In the display, notice that **Sasuser.Univ** contains several file location levels. The **Exponential_x** file location contains the **Exp** output object. When you double-click an output object, such as **Exp**, that output object is replayed in the Results window to all open destinations.

Display 6.1 Documents Window



A Documents window contains these items:

entry is an output object, link, or file location.

Note: Only output objects of the type *Document* are displayed in the Documents window. △

file location is a grouping of ODS document entries.

link is a symbolic link from one specified output object to another output object.

Note: Within the Documents window, a link is called a *shortcut*. △

ODS document is the name of an ODS document.

ODS Document Icon

The Results window and the Documents window use this icon to indicate an ODS document output object:

Display 6.2 ODS Document Icon

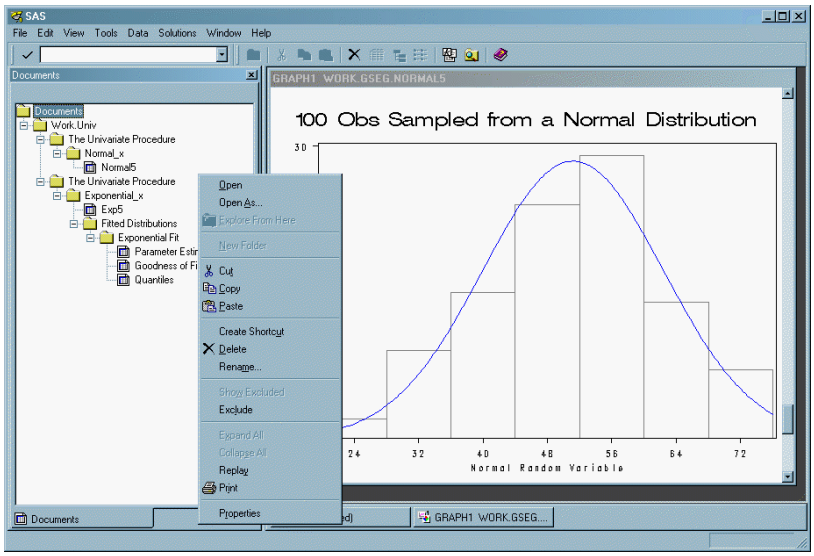
Operating Environment Information: The ODS Documents window on z/OS has the same functionality, but does not use graphical icons. △

Using the Documents Window Pop-up Menu

The Documents window has a pop-up menu with features that are also available through batch processing. To view the Documents window pop-up menu, follow these steps:

1. Type **odsdocuments** in the command bar. The Documents window appears.
2. Right-click any entry in the Documents window. The pop-up menu appears.

Display 6.3 Pop-up Menu for the Documents Window



The following table describes the pop-up menu item features. The availability of each pop-up menu item depends on which entry you select in the Documents window.

Table 6.3 Tasks You Can Do with the Documents Window Pop-up Menu *

Task	Menu Item
Open the selected object in the Results Viewer	Open
Select a new ODS destination output type	Open As
Open a window in tree view and list view	Explore From Here
Create a new folder	New Folder
Remove the selected entry from the Documents window	Cut
Copy the selected entry to system memory	Copy
Paste the copied entry to the selected location	Paste
Create a shortcut to the entry	Create Shortcut
Delete the selected entry	Delete

Task	Menu Item
Rename the selected entry	Rename
Show the entries that were previously excluded	Show Excluded
Remove from the tree, but do not delete the selected entry	Exclude
Expand all the levels of the tree	Expand All
Collapse all the levels in the tree	Collapse All
Replay the selected entry to all open ODS destinations	Replay
Print the selected entry	Print
Display the properties of the selected entry	Properties

* Available menu choices vary, depending on the selected entry.

ODS Documents in the Results Window

Understanding When to Use the Results Window

Although the Results window (like the Documents window) lists ODS documents, the Results window also lists other types of output objects, such as PDF and HTML. The Results window displays the following information:

- the output object types that are created when you run a SAS program in the current SAS session. SAS creates an output object for each ODS destination that was open at the time you executed a procedure during the current SAS session only.
- the results after you create a new output object from the Documents window using the **Open As** or **Replay** feature.
- the properties of an entry.

The results window also deletes or renames entries.

See “Comparisons between the Documents Window and the Results Window” on page 371.

Viewing Entries in the Results Window

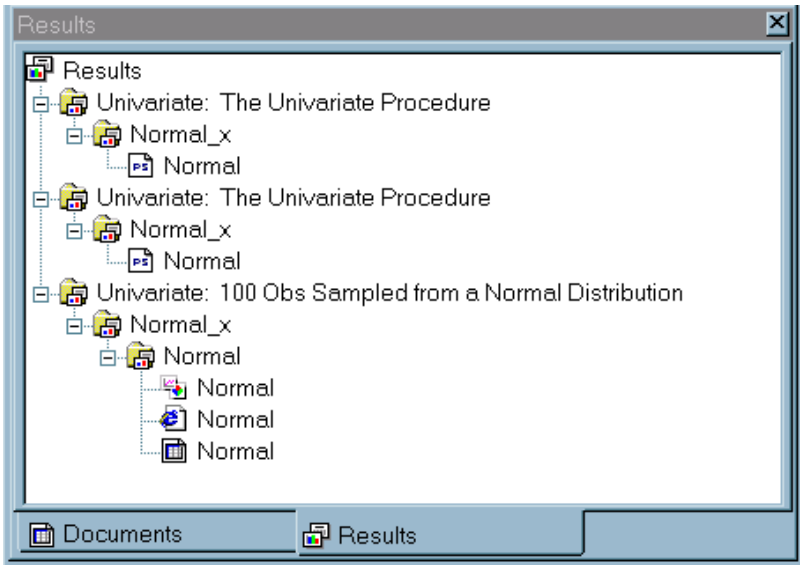
To view the Results window, submit this command in the command bar:

```
odsresults
```

You can also view the Results window by selecting: **View ▶ Results**

The following display shows the Results window with files and output objects. The last file is **Univariate:100 Obs Sampled from a Normal Distribution**. Under this file is the same output object sent to three different destinations. Each output object is named **Normal**, and the destinations are LISTING, HTML, and DOCUMENT.

Display 6.4 Results Window Showing the Output Object *Normal* in Three Formats



For more information about using the Results window, make the Results window the active window and select **Help ▸ Using This Window**.

Comparisons between the Documents Window and the Results Window

Table 6.4 Tasks That You Can and Cannot Do in the Documents Window and the Results Window

Task	Documents window	Results window
View all SAS documents including those stored in SAS libraries	Yes	Yes
View output object types that are created when you run a SAS program, such as HTML, PDF, and SAS document	No	Yes
View the results after you create a new output object	Yes	Yes
Customize the layout of output objects	Yes	No
View the property information of SAS documents	Yes	Yes
View the properties of an output object	No	Yes
Delete or rename entries	Yes	Yes
Copy or move SAS documents	Yes	No

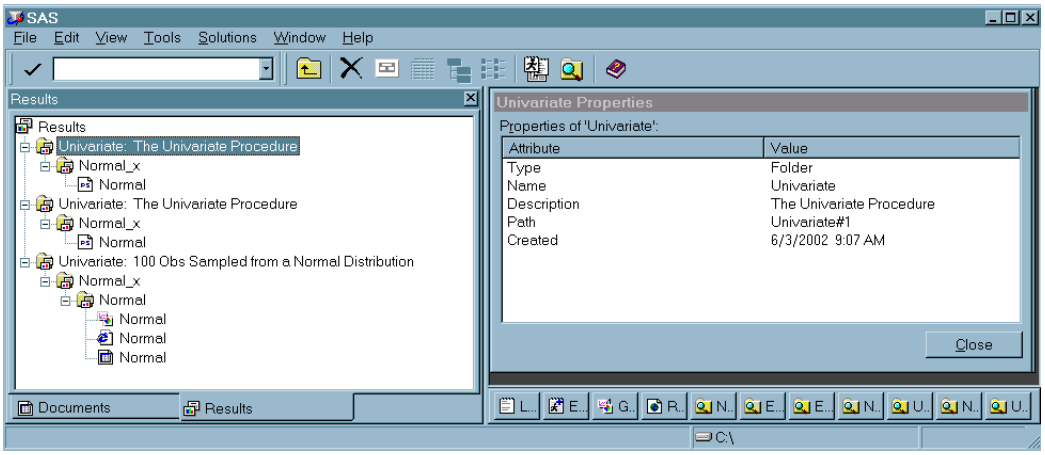
Task	Documents window	Results window
Create shortcuts to SAS documents	Yes	No
Drag and drop output objects	Yes	No

Viewing the Properties of an Entry

Any entry that you select in either the Results window or the Documents window has an associated Properties window. To view the properties of an entry, follow these steps:

1. Select an entry from either the Results Window or the Documents window.
2. Right-click the entry. A pop-up menu appears.
3. Select **`Properties`**. The Properties window for the entry appears.

Display 6.5 Entry Properties Window



Items will vary, depending on the entry that you select in the Documents or Results windows. The Properties window for an ODS document output object can contain these items:

Created	is the date that the entry was created.
Document	is the SAS filename where the entry is located. The filename is in the form of *libref.filename*
Document path	the location of the entry in the tree structure. If you move the entry to another location in the Documents window, then this path will change.
Modified	is the date that the entry was modified.
Name	is the name of the entry.
Path	is the storage location inside the document of the entry.
Type	is the classification of the entry.

Creating Shortcuts in the Documents Window

The Documents window pop-up menu provides you with a **`Create Shortcut`** option. Shortcut links are useful when you are creating output that uses the same entry in

more than one place. Instead of copying the entry to each location, consider using a shortcut. Shortcuts have these advantages:

- Because a shortcut is a link to the original entry, any changes that you make to the original entry will appear when you select the shortcut.
- A shortcut uses fewer computer resources.

To create a shortcut, do this:

1. Right-click an entry in the Documents window. A pop-up menu appears.
2. Select **`Create Shortcut`**. A new shortcut entry appears below the selected entry.

Comparisons between the Documents Window and the Document Procedure

Table 6.5 Tasks That You Can and Cannot Do in the Documents Window and with the DOCUMENT Procedure

Task	Documents Window	Document Procedure
Create a new ODS document	Yes	Yes
Create a new folder	Yes	Yes
Import a data set or graph segment	No	Yes
Copy folders or output objects	Yes	Yes
Move folders or output objects	Yes	Yes
Create a symbolic link from one output object to another output object	Yes	Yes
Delete a document, folder, or output object	Yes	Yes
Rename a folder or output object	Yes	Yes
Assign a description to a folder or output object	Yes	Yes
Prevent entries from being displayed when they are replayed	Yes	Yes
Show entries that are excluded	Yes	Yes
Enable hidden entries to be displayed	Yes	Yes
Replay to the specified open ODS destinations	Yes	Yes
Determine the path specification	Yes	Yes
Set or display the current directory	No	Yes

Task	Documents Window	Document Procedure
Create or delete a page break	No	Yes
Create or modify title lines	No	Yes
Create or modify subtitles	No	Yes
Create of modify the lines of text before output objects	No	Yes
Create or modify the lines of text after output objects	No	Yes
Create or modify footnote lines	No	Yes
Create text strings in the current folder	No	Yes

Examples: DOCUMENT Procedure

Example 1: Navigating the File Location and Listing the Entries

Procedure features:

ODS DOCUMENT statement option:

NAME=

DOC statement option:

NAME=

LIST statement options:

entry
LEVELS=
DETAILS

DIR statement option:

path

ODS destinations:

DOCUMENT
LISTING
HTML

Procedure output:

PROC DOCUMENT

Program Description

This example shows you how to do these tasks:

- name an ODS document
- see what ODS documents exist
- open a document for browsing or editing purposes

- list one or more entries
- change file locations

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The NONUMBER option suppresses the printing of page numbers.

```
options nodate nonumber;
```

Create the DistrData data set. The DistrData data set contains the statistical information that PROC UNIVARIATE uses to create the histograms.

```
data distrdata;
  drop n;
  label Normal_x='Normal Random Variable'
        Exponential_x='Exponential Random Variable';
  do n=1 to 100;
     Normal_x=10*rannor(53124)+50;
     Exponential_x=ranexp(18746363);
     output;
  end;
run;
```

Create the ODS document Univ and open the DOCUMENT destination. The ODS DOCUMENT statement opens the DOCUMENT destination. The NAME= option assigns the name Univ to the ODS document that contains the information from this PROC UNIVARIATE program. Note that by default Univ will be created in the WORK library. Assign a libref to create Univ in a permanent library.

```
ods document name=univ;
```

Create a normal distribution histogram. The TITLE statement specifies the title of the normal distribution histogram. The PROC UNIVARIATE step creates a normal distribution histogram from the DistrData data set.

```
title '100 Obs Sampled from a Normal Distribution';
proc univariate data=distrdata noprint;
  var Normal_x;

  histogram Normal_x /normal(noprint) cbarline=grey name='normal';
run;
```

Create an exponential distribution histogram. The TITLE statement specifies the title of the exponential histogram. The PROC UNIVARIATE step creates an exponential distribution histogram from the DistrData data set.

```
title '100 Obs Sampled from an Exponential Distribution';

proc univariate data=distrdata noprint;
  var Exponential_x;

  histogram /exp(fill l=3) cfill=yellow midpoints=.05 to 5.55 by .25
             name='exp';
run;
```

Close the DOCUMENT destination. If the DOCUMENT destination is not closed, no DOCUMENT procedure output can be viewed.

```
ods document close;
title;
```

View the ODS documents, choose an ODS document, and list the entries of the opened ODS document. The DOC statement (with no arguments specified) prints a listing of all of the available documents that are in the SAS System.

The DOC statement with the NAME= option specifies the current document, WORK.Univ. The LIST statement with the LEVELS=ALL option lists detailed information on all levels of the document WORK.Univ.

```
proc document;
   doc;
   doc name=univ;
   list/levels=all;
```

Set the path to EXPONENTIAL, list the contents of the EXPONENTIAL file location, select a table, and list the details of the table you selected. The DIR statement changes the current file location to **`univariate#2\exponential_x\fitteddistributions\exponential`**. The path **`univariate#2\exponential_x\fitteddistributions\exponential`** was obtained from the listing of the WORK.Univ document.

The LIST statement (with no arguments) lists the contents of **`EXPONENTIAL`** (see Display 6.8 on page 377). The LIST FITQUANTILES\DETAILS statement specifies that ODS opens the FitQuantiles table and lists its details (see Display 6.9 on page 377).

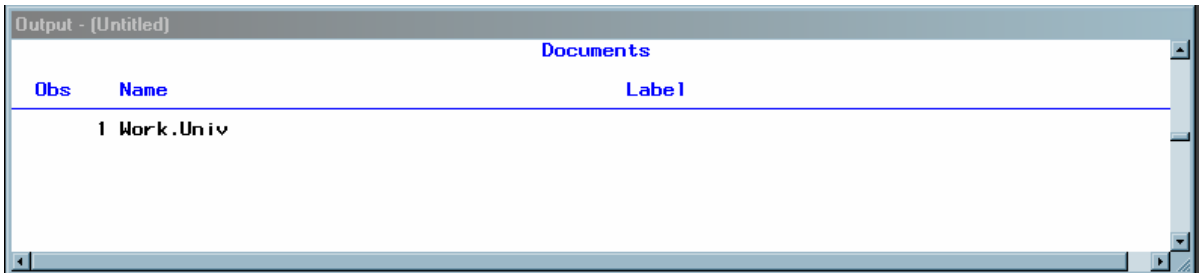
```
   dir univariate#2\exponential_x\fitteddistributions\exponential;
   list;
   list fitquantiles/details;
run;
```

Terminate the DOCUMENT procedure. Specify the QUIT statement to terminate the DOCUMENT procedure. If you omit QUIT, then you will not be able to view DOCUMENT procedure output.

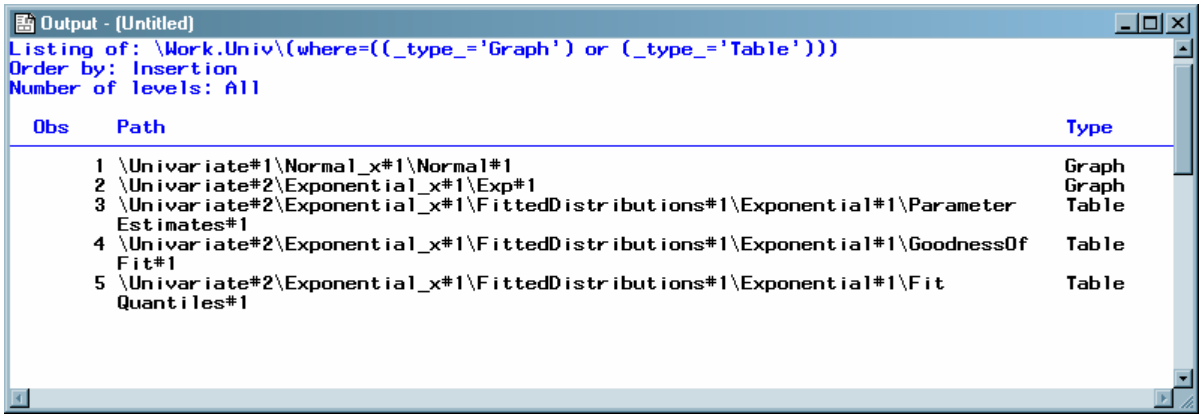
```
quit;
```

Output

Display 6.6 Current ODS Document in Output



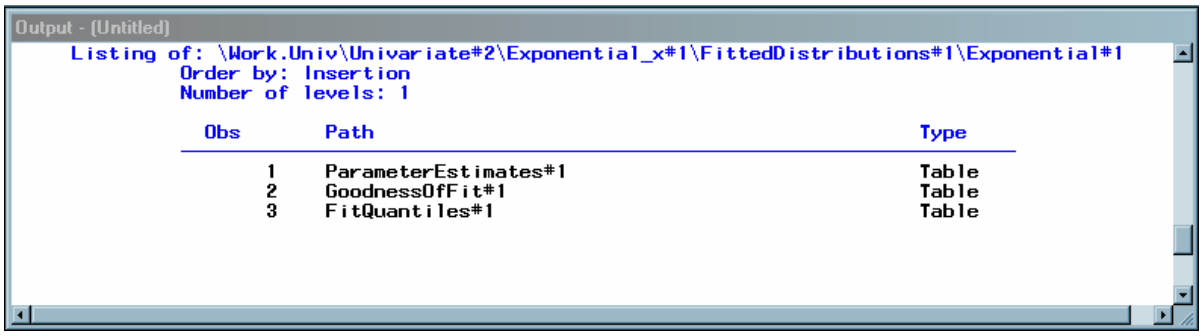
Display 6.7 List of the Entries of the ODS Document WORK.Univ and the Properties of Those Entries



Output - (Untitled)

```
Listing of: \Work.Univ\(where=((_type_='Graph') or (_type_='Table')))
Order by: Insertion
Number of levels: All

 Obs    Path                                                                    Type

   1 \Univariate#1\Normal_x#1\Normal#1                                         Graph
   2 \Univariate#2\Exponential_x#1\Exp#1                                       Graph
   3 \Univariate#2\Exponential_x#1\FittedDistributions#1\Exponential#1\Parameter Table
     Estimates#1
   4 \Univariate#2\Exponential_x#1\FittedDistributions#1\Exponential#1\GoodnessOf Table
     Fit#1
   5 \Univariate#2\Exponential_x#1\FittedDistributions#1\Exponential#1\Fit       Table
     Quantiles#1
```

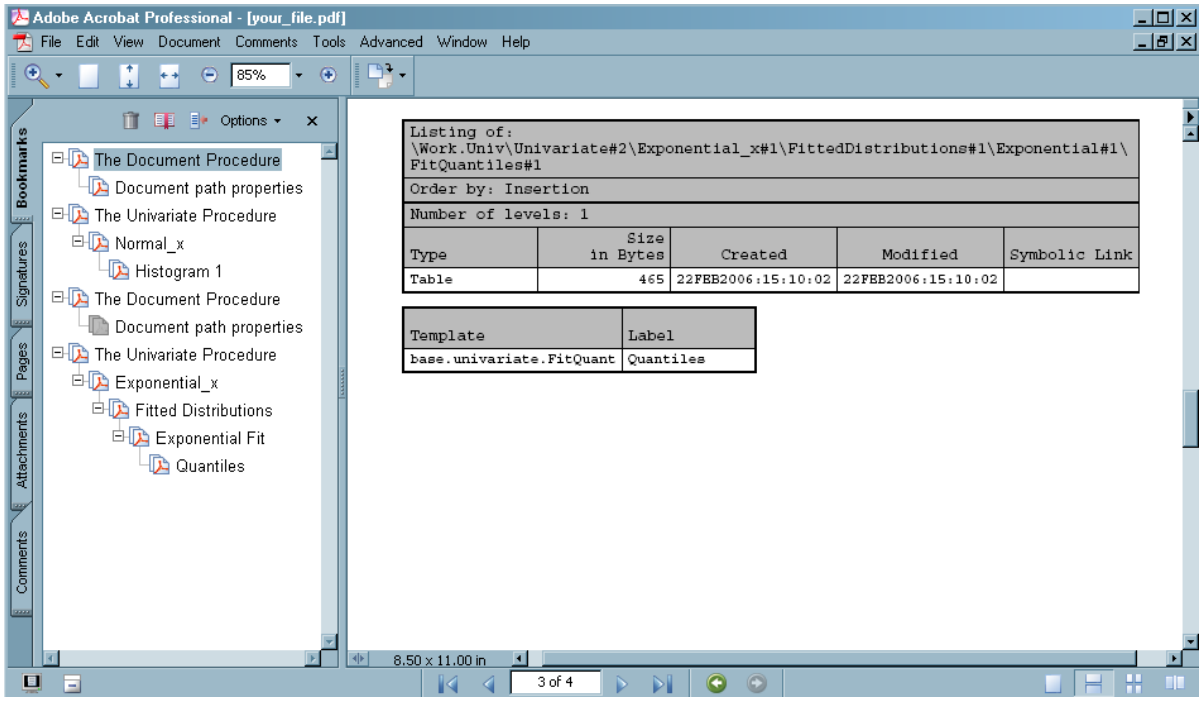
Display 6.8 List of the Entries of the Exponential#1 Entry and the Properties of Those Entries



Output - (Untitled)

```
Listing of: \Work.Univ\Univariate#2\Exponential_x#1\FittedDistributions#1\Exponential#1
Order by: Insertion
Number of levels: 1

 Obs    Path                                                                    Type

   1    ParameterEstimates#1                                                    Table
   2    GoodnessOfFit#1                                                         Table
   3    FitQuantiles#1                                                          Table
```

Display 6.9 Details of the FitQuantiles#1 Table



Adobe Acrobat Professional - [your_file.pdf]

File Edit View Document Comments Tools Advanced Window Help

85%

Options

Bookmarks Signatures Pages Attachments Comments

- The Document Procedure
  - Document path properties
- The Univariate Procedure
  - Normal_x
    - Histogram 1
- The Document Procedure
  - Document path properties
- The Univariate Procedure
  - Exponential_x
    - Fitted Distributions
      - Exponential Fit
        - Quantiles

Listing of: \Work.Univ\Univariate#2\Exponential_x#1\FittedDistributions#1\Exponential#1\FitQuantiles#1

Order by: Insertion

Number of levels: 1

Type	Size in Bytes	Created	Modified	Symbolic Link
Table	465	22FEB2006:15:10:02	22FEB2006:15:10:02	

Template	Label
base.univariate.FitQuant	Quantiles

8.50 x 11.00 in

3 of 4

Example 2: Opening and Listing ODS Documents

Procedure features:
PROC DOCUMENT statement option:
NAME=
DIR statement
LIST statement options:
DETAILS
LEVELS
where-expression
REPLAY statement

ODS destinations:
DOCUMENT
LISTING
PDF

Procedure output:
PROC DOCUMENT
PROC UNIVARIATE

Data set: See "DistrData on page 375"

ODS document: See "Creating the Univ ODS Document" on page 872

Program Description

This example shows you how to do these tasks:

- open an ODS document
- replay a table and send the output to the LISTING and PDF destinations
- list specific entries in an ODS document by using WHERE expressions
- change file locations
- list the details of a specified entry
- replay an ODS document to a PDF file

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The NONUMBER option suppresses the printing of page numbers.

```
options nodate nonumber;
```

Open the ODS document WORK.Univ. The PROC DOCUMENT statement with the NAME= option specified opens the ODS document WORK.Univ, which was created in Example 1 on page 374, for updates.

```
proc document name=univ;
```

Specify that you want to replay the output to a PDF file. The ODS PDF statement opens the PRINTER destination and replays the histogram to the PDF destination. The FILE= statement sends all output objects to the external file that you specify.

```
ods pdf file= 'your_file.pdf';
```

List the entries that are associated with the current document and replay a histogram.

By using a WHERE expression, the LIST statement lists only entries that are graphs or tables. The LEVELS=ALL option specifies that detailed information on all levels be shown. The ^ symbol represents the current path.

The REPLAY statement replays the Normal#1 entry to all open ODS destinations.

```
    list ^ (where=(_type_ = 'Graph' or _type_ = 'Table') ) /levels=all;
    replay univariate#1\Normal_x#1\Normal#1;
```

Change the current file location, list the details of the FitQuantiles table, and replay the FitQuantiles table. The DIR statement changes the current file location to **`univariate#2\exponential_x\fitteddistributions\exponential#1`**.

The LIST statement with the DETAILS option specifies the listing of the properties of the entry FitQuantiles table.

The REPLAY statement replays FITQUANTILES to open destinations.

```
    dir univariate#2\exponential_x\fitteddistributions\exponential#1;
    list fitquantiles/details;
    replay fitquantiles;
run;
```

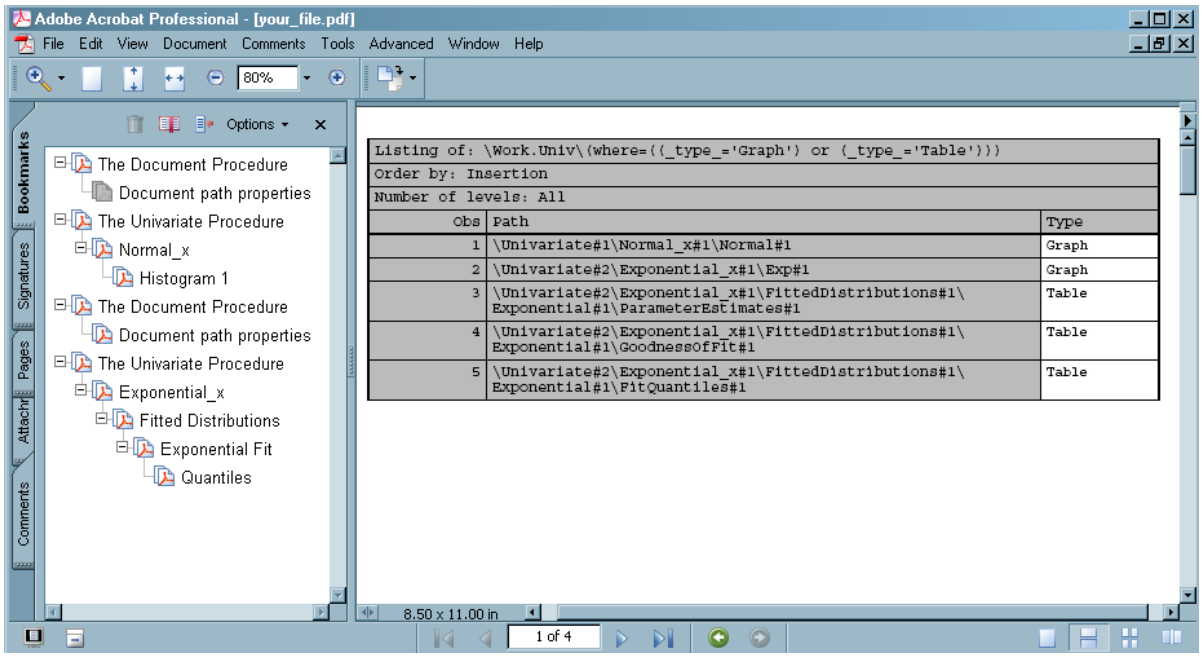
Terminate the DOCUMENT procedure and close the PDF destination. Specify the QUIT statement to terminate the DOCUMENT procedure. If you omit QUIT, then you will not be able to view DOCUMENT procedure output. The ODS PDF CLOSE statement closes the PDF destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```
quit;
ods pdf close;
```

Output

Display 6.10 List of the Graphs and Tables Found in WORK.Univ, Viewed in Acrobat Reader

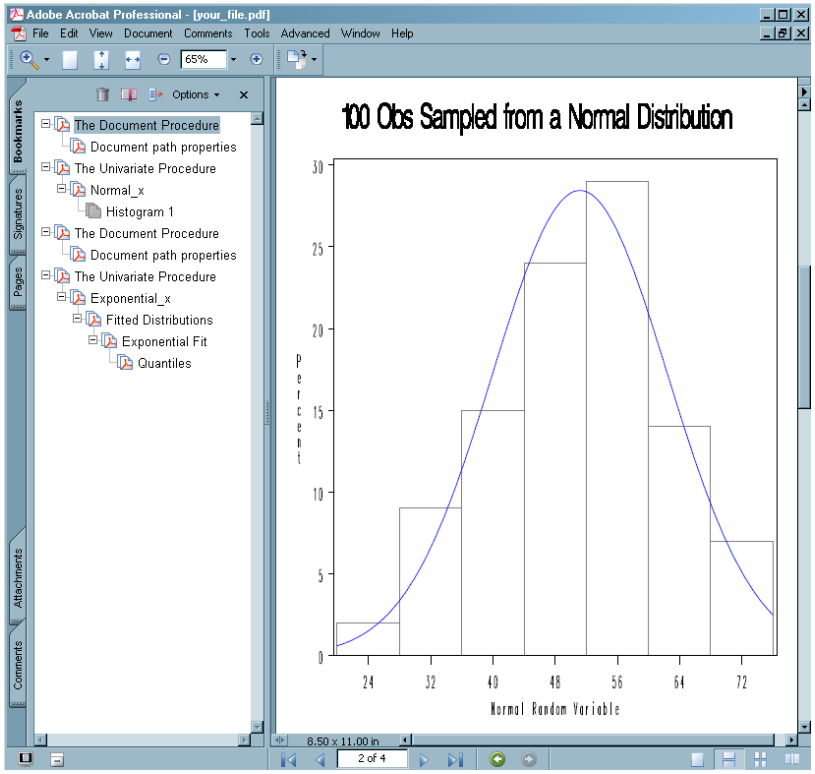
This display is page 1 of the ODS document WORK.Univ that was sent to the PDF destination. You can browse the output by clicking the bookmarks.



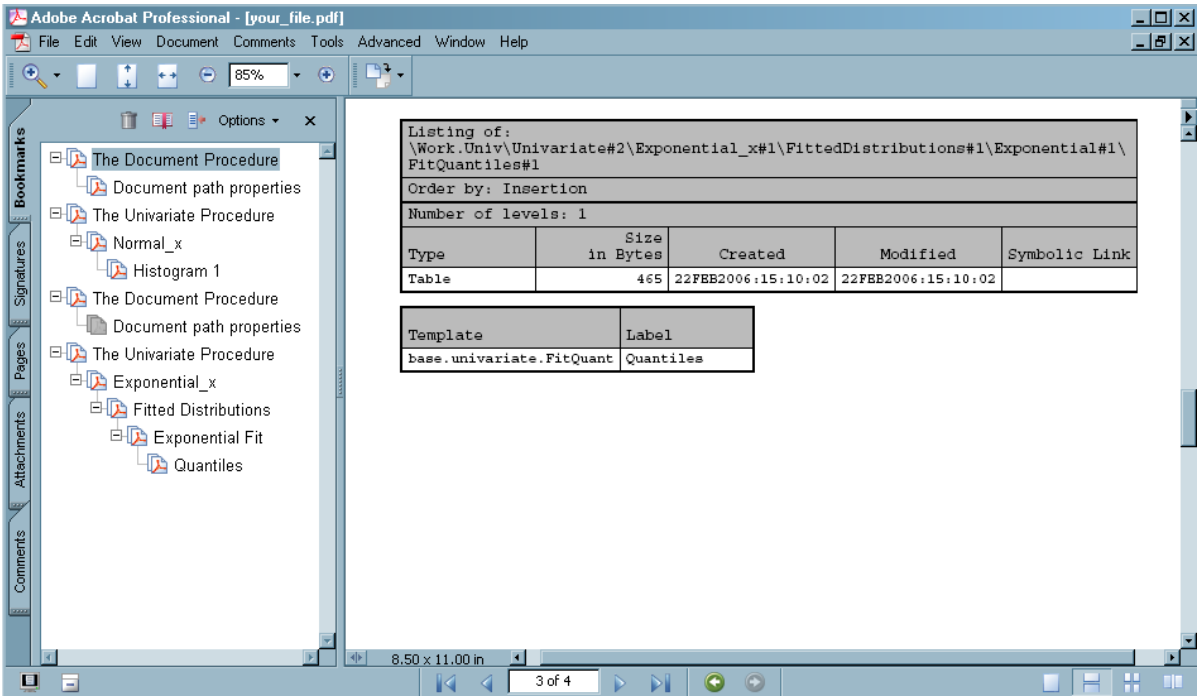
Listing of: \Work.Univ\(where=((_type_='Graph') or (_type_='Table')))
Order by: Insertion
Number of levels: All

Obs	Path	Type
1	\Univariate#1\Normal_x#1\Normal#1	Graph
2	\Univariate#2\Exponential_x#1\Exp#1	Graph
3	\Univariate#2\Exponential_x#1\FittedDistributions#1\ Exponential#1\ParameterEstimates#1	Table
4	\Univariate#2\Exponential_x#1\FittedDistributions#1\ Exponential#1\GoodnessOfFit#1	Table
5	\Univariate#2\Exponential_x#1\FittedDistributions#1\ Exponential#1\FitQuantiles#1	Table

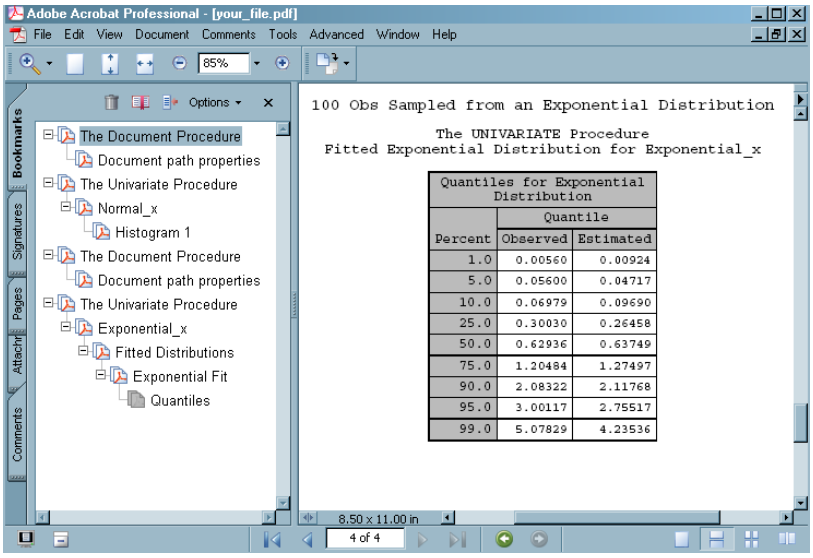
Display 6.11 Replayed Normal Distribution Histogram



Display 6.12 Details of the FitQuantiles#1 Table



Display 6.13 Replayed FitQuantiles#1 Table



Example 3: Managing Entries

Procedure features:

PROC DOCUMENT statement option:

NAME=

DIR statement

LIST statement option:
LEVELS=
NOTE statement
OBANOTE statement
OBBNOTE statement
OBFOOTN statement
OBPAGE statement
OBSTITLE statement
OBTITLE statement
REPLAY statement

ODS destinations:
DOCUMENT
HTML
LISTING

Procedure output:
PROC CONTENTS

Program Description

This example shows you how to do these tasks:

- generate PROC CONTENTS output to the DOCUMENT destination
- change the title and footnote of the output
- add object footer and object heading notes to the output
- change the subtitle of the output
- add a note to the document
- add a page break to the output

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The PAGENO= option specifies the starting page number.

```
options nodate pageno=1;
```

Close the LISTING destination and open the DOCUMENT destination. The NAME= option creates an ODS document named Class.

```
ods listing close;
ods document name=class;
```

Specify a global title and footnote. The TITLE statement creates a title that is used until you change it with another statement. The FOOTNOTE statement creates a footnote that is used until you change it with another statement.

```
title 'Title Specified by the Global TITLE Statement';
footnote 'Footnote Specified by the Global FOOTNOTE Statement';
```

View the contents of the SAS data set. The CONTENTS procedure shows the contents of the SAS data set SASHELP.Class.

```
proc contents data=sashelp.class;
run;
```

Close the DOCUMENT destination and create LISTING output. The entries in the ODS document Class are used in the remainder of this example. The ODS LISTING statement opens the LISTING destination and creates listing output.

```
ods document close;
ods listing;
```

Change the global title.

- The OBTITLE statement assigns a new title to the Attributes#1 entry. See Display 6.14 on page 386
- The NAME= option specifies the current ODS document.
- The LIST statement with the LEVELS=ALL option shows a list of entries in the Class document. Note that PROC DOCUMENT is still running after the RUN statement executes.
- The DIR statement changes the current path to **\Contents#1\DataSet#1**.
- The QUIT statement terminates PROC DOCUMENT.

```
proc document name=class;
   list /levels=all;
run;
   dir \Contents#1\DataSet#1;
run;
   obtitle Attributes#1 'Title Specified by the OBTITLE Statement';
run;
quit;
```

Add an object heading note to the output.

- The OBBNOTE statement assigns an object heading note to the Attributes#1 entry. See Display 6.14 on page 386
- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to **\Contents#1\DataSet#1**.
- The QUIT statement terminates PROC DOCUMENT.

```
proc document name=class;
   dir \Contents#1\DataSet#1;
run;
   obbnote Attributes#1 'Object Heading Note Specified by the OBBNOTE Statement';
run;
quit;
```

Change the global footnote.

- The OBFOOTN statement assigns a new footnote to the Variables#1 entry.
- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to **\Contents#1\DataSet#1**.
- The QUIT statement terminates PROC DOCUMENT.

```
proc document name=class;
   dir \Contents#1\DataSet#1;
run;
   obfootn Variables#1 'Change the Global Footnote with the OBFOOTN Statement';
run;
quit;
```

Add an object footer note

- The OBANOTE statement assigns an object footer note to the Attributes#1 entry. See Display 6.14 on page 386
- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to **\Contents#1\DataSet#1**.
- The QUIT statement terminates PROC DOCUMENT.

```
proc document name=class;
   dir \Contents#1\DataSet#1;
run;
   obanote Attributes#1 'Object Footer Note Specified by the OBANOTE Statement';
run;
quit;
```

Change the subtitle of the output.

- The OBSTITLE statement changes the subtitle. The subtitle identifies the procedure that produced the output. See Display 6.14 on page 386
- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to **\Contents#1\DataSet#1**.
- The QUIT statement terminates PROC DOCUMENT.

```
proc document name=class;
   dir \Contents#1\DataSet#1;
run;
   obstitle Attributes#1 'Subtitle Specified by the OBSTITLE Statement';
run;
quit;
```

Add a note to the document.

- The NOTE statement adds a note object named ADDNOTE to the ODS document.
- The NAME= option specifies the current ODS document.
- The LIST statement with the LEVELS=ALL option shows a list of entries in the Class document.
- The QUIT statement terminates PROC DOCUMENT.

```
proc document name=class;
   note addnote 'Note added to the document';
```

```
   list /levels=all;
run;
quit;
```

Add a page break to the output, create HTML output, and replay Variables#1.

- The ODS HTML statement opens the HTML destination and creates HTML 4.0 output.
- The STYLE= option specifies that ODS use the style D3D.
- The OBPAGE statement inserts a page break.
- The NAME= option specifies the current ODS document.
- The REPLAY statement replays the Variables#1 object and generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.

```
ods html file='your_file.html' style=d3d;
proc document name=class;
   obpage \Contents#1\DataSet#1\Variables#1;
   replay;
run;
quit;
```

Close the HTML and LISTING destinations. The ODS _ALL_ CLOSE statement closes all open ODS output destinations so that you can view the output.

```
ods _all_ close;
```

Output

Display 6.14 Global Title, Global Footnote, Subtitle, Object Heading Note, Object Footer Note, and Note

Title Specified by the OBTITLE Statement

Subtitle Specified by the OBSTITLE Statement

Object Heading Note Specified by the OBBNOTE Statement

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Wednesday, May 12, 2004 10:53:55 PM	Observation Length	40
Last Modified	Wednesday, May 12, 2004 10:53:55 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Object Footer Note Specified by the OBANOTE Statement

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
File Name	C:\Program Files\SAS\SAS 9.1\core\sashelp\class.sas7bdat
Release Created	9.0101M3
Host Created	XP_PRO

Change the Global Footnote with the OBFOOTN Statement

Title Specified by the Global TITLE Statement

The CONTENTS Procedure

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Note added to the document

Example 4: Listing BY-Group Entries

Procedure features:

LIST statement options:

BYGROUPS
LEVELS
LIST

PROC DOCUMENT statement option:

NAME=

OBTEMPL statement

ODS destinations:

DOCUMENT

LISTING

Procedure output:

PROC DOCUMENT

PROC STANDARD

Program Description

This example shows you how to do these tasks:

- generate PROC STANDARD output to the DOCUMENT destination
- view the table template that describes how to display the PROC STANDARD output
- create an ODS document
- open an ODS document
- list the BY group entries in an ODS document

Program

Set the SAS system options, create the ODS document MyDocument, and open the DOCUMENT destination. The NODATE option suppresses the display of the date and time in the output. The NONUMBER option suppresses the printing of page numbers. The ODS DOCUMENT statement with the NAME= option specified opens the ODS document MyDocument and provides Write access as well as Read access. Note that by default MyDocument will be created in the WORK library. Assign a libref to create MyDocument in a permanent library.

```
options nodate nonumber;
ods document name=mydocument(write);
```

Create and sort the Score data set. This data set contains test scores for students who took two tests and a final exam. The SORT procedure sorts the data set by the BY variables Section and Student.

```
data score;
   input Student Section Test1-Test3;
   stest1=test1;
   stest2=test2;
   stest3=test3;
```

```
   datalines;
238900545 1 94 91 87
254701167 1 95 96 97
238806445 2 91 86 94
999002527 2 80 76 78
263924860 1 92 40 85
459700886 2 75 76 80
416724915 2 66 69 72
999001230 1 82 84 80
242760674 1 75 76 70
990001252 2 51 66 91
;
run;

proc sort data=score;
   by Section Student;
run;
```

Generate the standardized data and create the output data set StndScore. PROC STANDARD uses a mean of 80 and a standard deviation of 5 to standardize the values. OUT= identifies StndScore as the data set to contain the standardized values. The PRINT option prints the statistics. The ODS LISTING statement closes the listing output so that no output will be viewed.

```
ods listing close;
proc standard mean=80 std=5 out=StndScore print;
```

Create the standardized values for each BY group and specify the variables to standardize. The BY statement standardizes the values separately by section number and student id. The VAR statement specifies the variables to standardize and their order in the output.

```
   by section student;
   var stest1-stest3;
run;
```

Close the DOCUMENT destination. If the DOCUMENT destination is not closed, no DOCUMENT procedure output can be viewed.

```
ods document close;
```

Open the ODS document MyDocument, list the entries, and view the table template that determines how the PROC STANDARD output will display. The PROC DOCUMENT statement with the NAME= option specified opens the ODS document WORK.MyDocument. The LIST statement with the LEVELS=ALL option lists detailed information on all levels of the document WORK.MyDocument. The BYGROUPS option creates columns in the list statement output for BY group information. The names of the columns with the BY group information are the names of the BY variables, Section and Student. To see what the output will look like if you omit the BYGROUPS option, see Display 6.15 on page 389. The OBTEMPL statement writes the table template that is associated with the output object Standard#1 to the listing destination. The ODS DOCUMENT CLOSE statement closes the DOCUMENT destination. If the DOCUMENT destination is not closed, no DOCUMENT procedure output can be viewed.

Note: If you omit LEVELS=ALL, then no entry list will be created. This is because ODS cannot find any BY groups at the directory level and only BY groups are listed when the BYGROUPS option is specified △

```
ods listing;
proc document name=mydocument;
   list/ levels=all bygroups;
   obtempl \Standard#1\ByGroup1#1\Standard#1;
run;
```

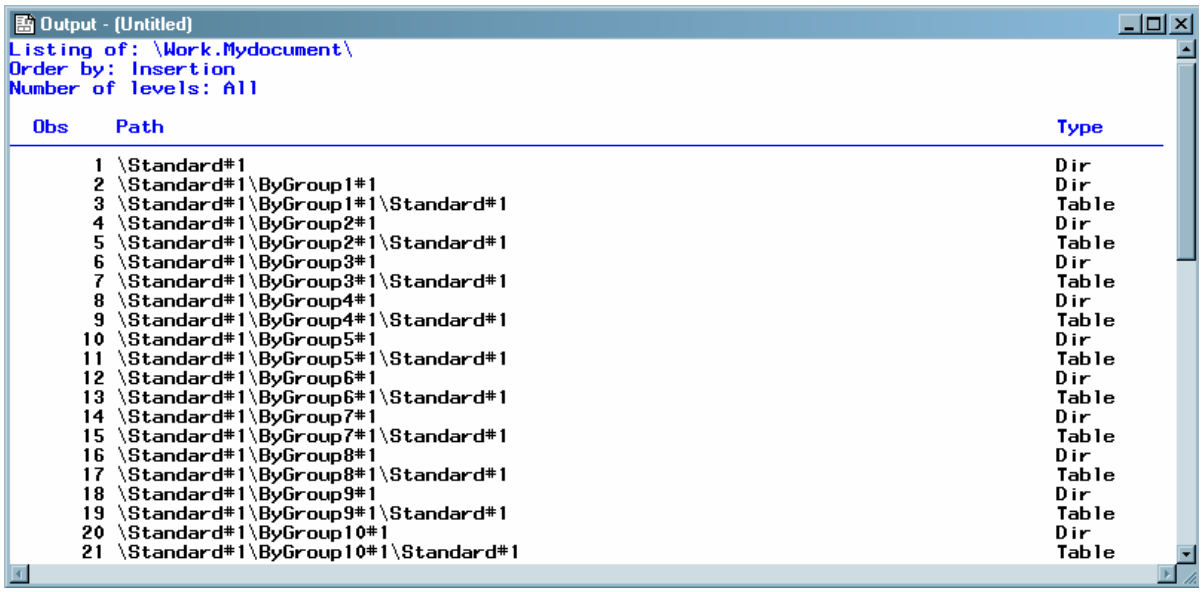
Terminate the DOCUMENT procedure. Specify the QUIT statement to terminate the DOCUMENT procedure. If you omit QUIT, then you will not be able to view DOCUMENT procedure output.

```
quit;
```

Output

Display 6.15 Listing of WORK.MyDocument without the BYGROUPS Option Specified

Without the BYGROUPS option specified, there are only three columns for this output: Obs, Path, and Type. All levels and all entries of WORK.MyDocument are displayed.

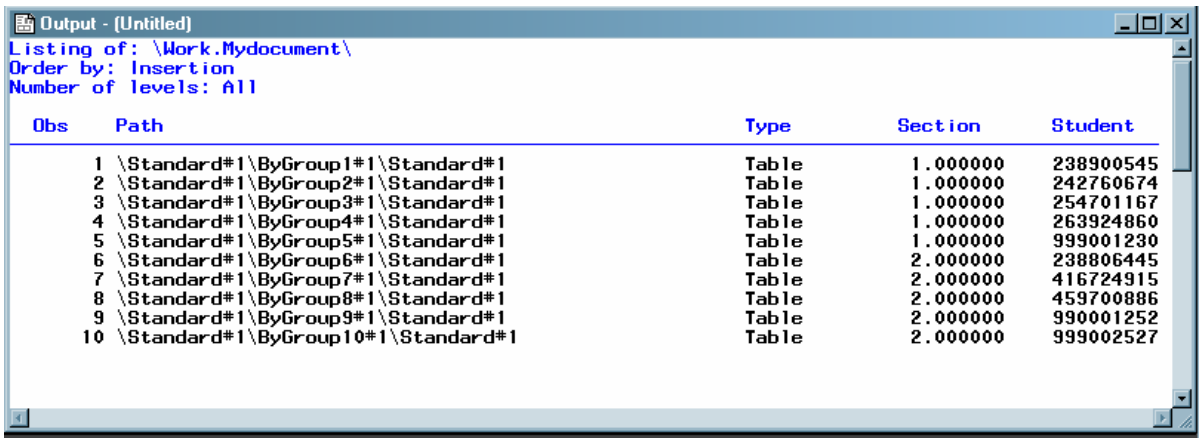

Output - (Untitled)

Listing of: \Work.Mydocument\
Order by: Insertion
Number of levels: All

Obs	Path	Type
1	\Standard#1	Dir
2	\Standard#1\ByGroup1#1	Dir
3	\Standard#1\ByGroup1#1\Standard#1	Table
4	\Standard#1\ByGroup2#1	Dir
5	\Standard#1\ByGroup2#1\Standard#1	Table
6	\Standard#1\ByGroup3#1	Dir
7	\Standard#1\ByGroup3#1\Standard#1	Table
8	\Standard#1\ByGroup4#1	Dir
9	\Standard#1\ByGroup4#1\Standard#1	Table
10	\Standard#1\ByGroup5#1	Dir
11	\Standard#1\ByGroup5#1\Standard#1	Table
12	\Standard#1\ByGroup6#1	Dir
13	\Standard#1\ByGroup6#1\Standard#1	Table
14	\Standard#1\ByGroup7#1	Dir
15	\Standard#1\ByGroup7#1\Standard#1	Table
16	\Standard#1\ByGroup8#1	Dir
17	\Standard#1\ByGroup8#1\Standard#1	Table
18	\Standard#1\ByGroup9#1	Dir
19	\Standard#1\ByGroup9#1\Standard#1	Table
20	\Standard#1\ByGroup10#1	Dir
21	\Standard#1\ByGroup10#1\Standard#1	Table

Display 6.16 Listing of WORK.MyDocument with the BYGROUPS Option Specified

With the BYGROUPS option specified there are now five columns. The additional columns, named Section and Student, were created by the BYGROUPS option. The BY variable names become the names of the columns. Only the entries containing BY group information are displayed. The entries that are directories are not displayed because they do not contain any actual BY group information.



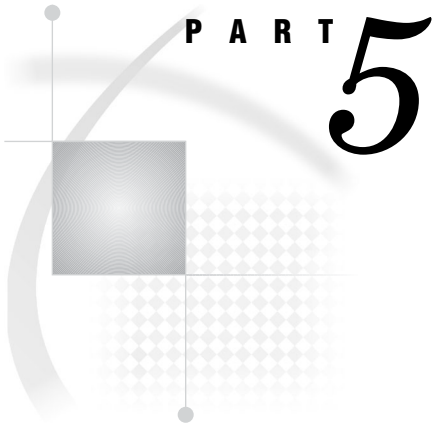
Output - (Untitled)

```
Listing of: \Work.Mydocument\
Order by: Insertion
Number of levels: All
```

Obs	Path	Type	Section	Student
1	\Standard#1\ByGroup1#1\Standard#1	Table	1.000000	238900545
2	\Standard#1\ByGroup2#1\Standard#1	Table	1.000000	242760674
3	\Standard#1\ByGroup3#1\Standard#1	Table	1.000000	254701167
4	\Standard#1\ByGroup4#1\Standard#1	Table	1.000000	263924860
5	\Standard#1\ByGroup5#1\Standard#1	Table	1.000000	999001230
6	\Standard#1\ByGroup6#1\Standard#1	Table	2.000000	238806445
7	\Standard#1\ByGroup7#1\Standard#1	Table	2.000000	416724915
8	\Standard#1\ByGroup8#1\Standard#1	Table	2.000000	459700886
9	\Standard#1\ByGroup9#1\Standard#1	Table	2.000000	990001252
10	\Standard#1\ByGroup10#1\Standard#1	Table	2.000000	999002527

Output 6.1 Listing View of the Table Template Associated with PROC STANDARD Output

```
proc template;
   define table Base.Standard;
      notes "Table template for PROC Standard.";
      column name mean std n label;

      define name;
         header = "Name";
         varname = Name;
         style = RowHeader;
      end;

      define mean;
         header = "Mean";
         format = D12.;
         varname = Mean;
      end;

      define std;
         header = "/Standard/Deviation";
         format = D12.;
         varname = stdDev;
      end;

      define n;
         header = "N";
         format = best.;
      end;

      define label;
         header = "Label";
         varname = Label;
      end;
      required_space = 3;
      byline;
      wrap;
   end;
run;
```



PART 5

The TEMPLATE Procedure

CHAPTER 7

TEMPLATE Procedure: Overview

Introduction to the TEMPLATE Procedure

Using the TEMPLATE Procedure

The TEMPLATE procedure enables you to customize the appearance of your SAS output. For example, you can create, extend, or modify existing templates for various types of output:

- styles
- tables
- crosstabulation tables
- columns
- headers
- footers
- tagsets
- ODS Graphics

ODS then uses these templates to produce formatted output.

You can also use the TEMPLATE procedure to navigate and manage the templates stored in template stores. Here are some tasks that you can do with PROC TEMPLATE:

- edit an existing template
- create links to an existing template
- change the location where you write new templates
- search for existing templates
- view the source code of a template

What Can You Do with the TEMPLATE Procedure?

Modify a Table Template That a SAS Procedure Uses

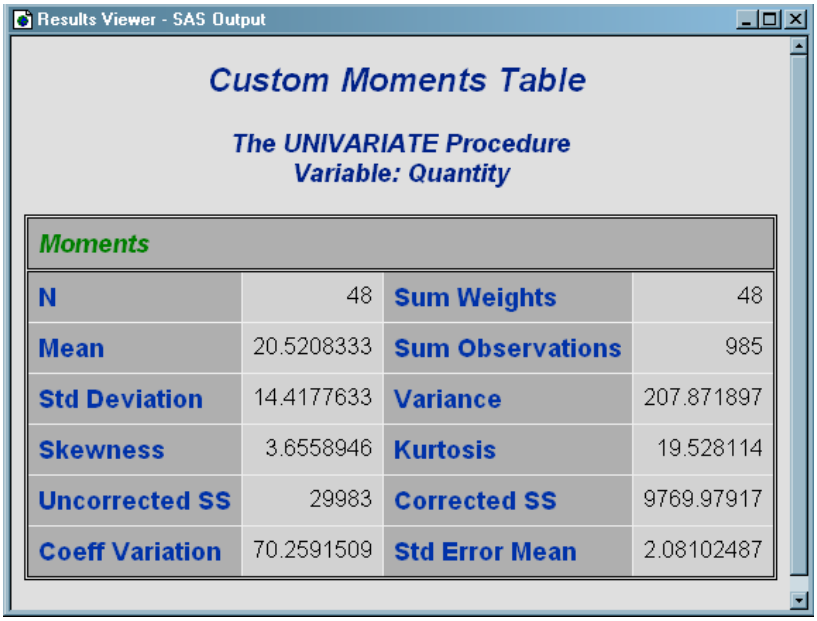
This output shows the use of a customized table template for the Moments output object from PROC UNIVARIATE. The program used to create the modified table template does the following:

- creates and edits a copy of the default table template
- edits a header within the table template
- sets column attributes to enhance the appearance of both the HTML and the Listing output

Output 7.1 Listing Output (Customized Moments Table) from PROC UNIVARIATE

```
                        Custom Moments Table                          1

                     The UNIVARIATE Procedure
   Variable:  CityPop_90  (1990 metropolitan pop in millions)

Moments
-----------------------------------------------------------------
N                           51    Sum Weights                 51

Mean                3.87701961    Sum Observations       197.728

Std Deviation       5.16465302    Variance            26.6736408

Skewness            2.87109259    Kurtosis             10.537867

Uncorrected SS      2100.27737    Corrected SS        1333.68204

Coeff Variation      133.21194    Std Error Mean      0.72319608
-----------------------------------------------------------------
```

Display 7.1 Customized HTML Output (Customized Moments Table) from PROC UNIVARIATE (Viewed with Microsoft Internet Explorer)



Moments			
N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Modify a Style

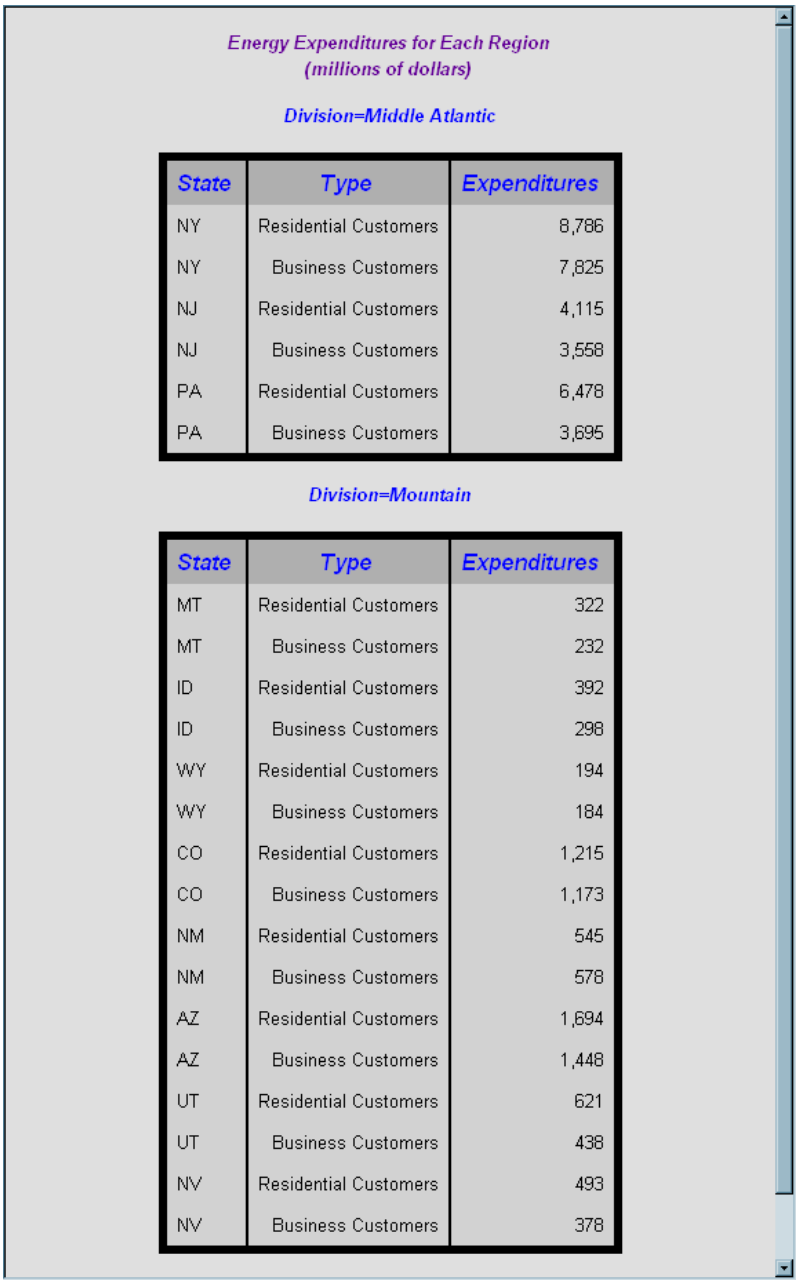
When you are working with styles (definitions), you are more likely to modify a style that SAS supplies than to write a completely new style. The following output uses the Styles.Default template that SAS provides, but includes changes made to the style in order to customize the output's appearance. Display 7.2 on page 398 shows changes made to both the contents file and the body file in the HTML output. In the contents file, the modified style makes changes to the following:

- the text of the header and the text that identifies the procedure that produced the output
- the colors for some parts of the text
- the font size for some parts of the text
- the spacing in the list of entries in the table of contents

In the body file, the modified style makes changes to the following:

- two of the colors in the color list. One of these colors is used as the foreground color for the table of contents, the byline, and column headings. The other is used for the foreground of many parts of the body file, including SAS titles and footnotes.
- the font size for titles and footnotes
- the font style for headers
- the presentation of the data in the table by changing attributes like cellspacing, rules, and borderwidth

Display 7.2 HTML Output (Viewed with Microsoft Internet Explorer)



Energy Expenditures for Each Region
(millions of dollars)

Division=Middle Atlantic

State	Type	Expenditures
NY	Residential Customers	8,786
NY	Business Customers	7,825
NJ	Residential Customers	4,115
NJ	Business Customers	3,558
PA	Residential Customers	6,478
PA	Business Customers	3,695

Division=Mountain

State	Type	Expenditures
MT	Residential Customers	322
MT	Business Customers	232
ID	Residential Customers	392
ID	Business Customers	298
WY	Residential Customers	194
WY	Business Customers	184
CO	Residential Customers	1,215
CO	Business Customers	1,173
NM	Residential Customers	545
NM	Business Customers	578
AZ	Residential Customers	1,694
AZ	Business Customers	1,448
UT	Residential Customers	621
UT	Business Customers	438
NV	Residential Customers	493
NV	Business Customers	378

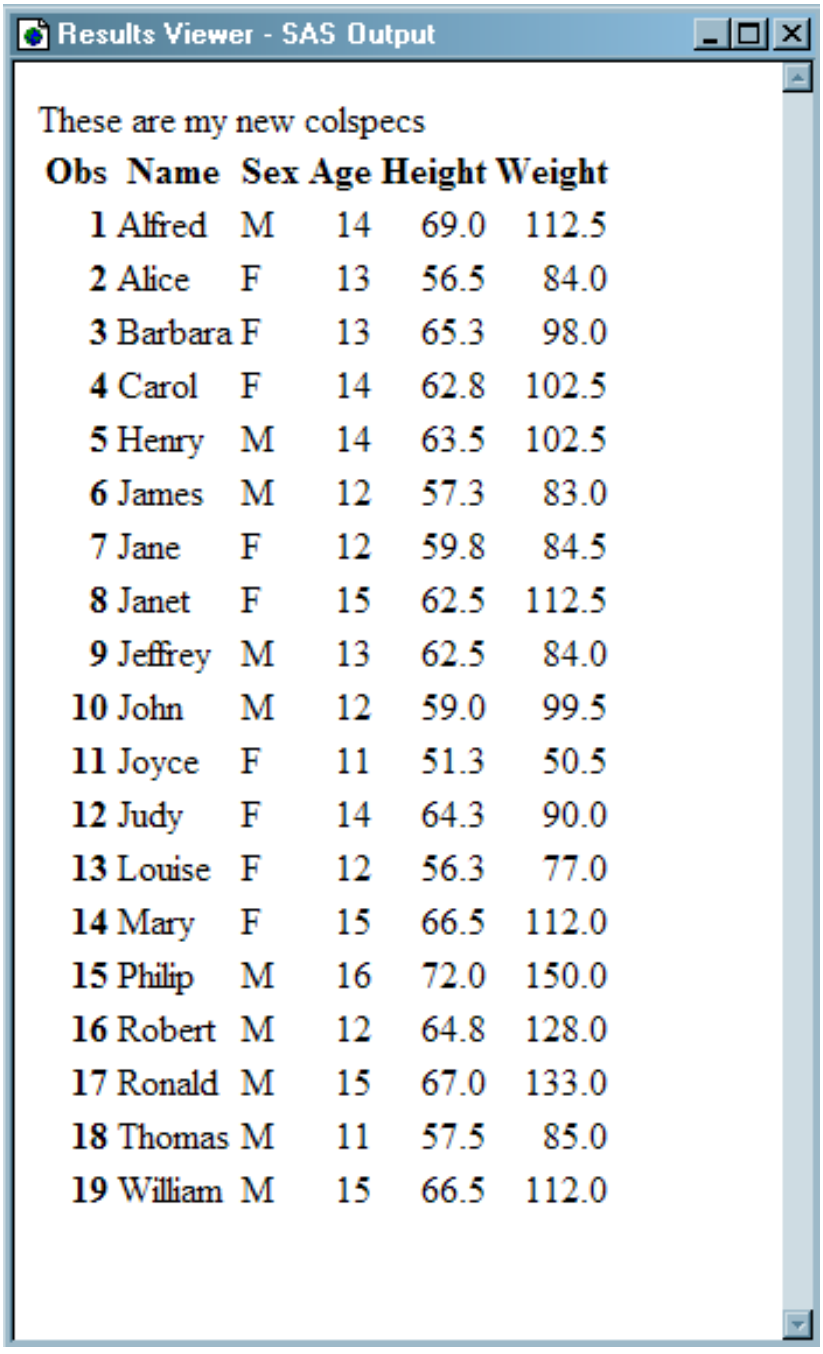
Create Your Own Tagset

Tagsets are used to create custom markup. You can create your own tagsets, extend existing tagsets, or modify a tagset that SAS supplies. This display shows the results from a new tagset **`TAGSET.MYTAGS`**.

Display 7.3 MYTAGS.CHTML Output (Viewed with Microsoft Internet Explorer)

To see the customized CHTML tagset, view the source from your web browser:

□ Select **View ► Source** from your browser's toolbar.



Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Create a Template-Based Graph

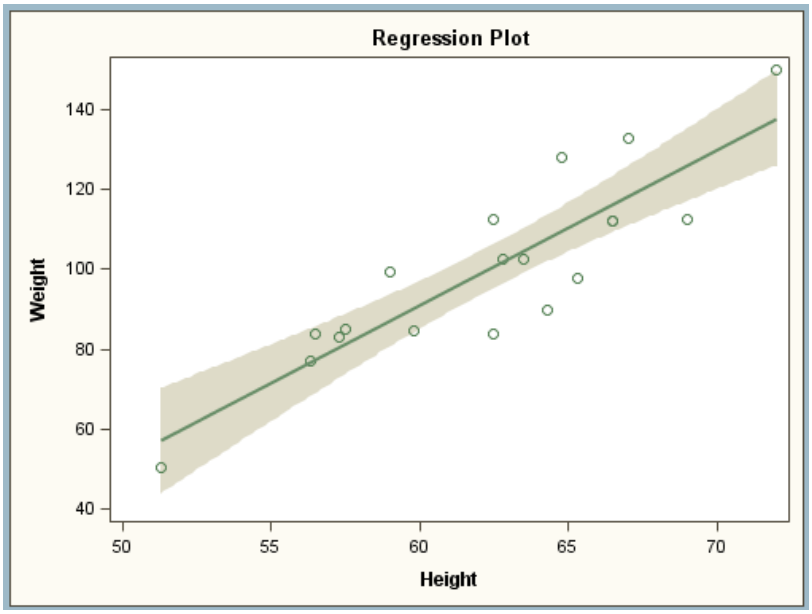
STATGRAPH templates are used to create output called ODS Graphics. For complete information see *SAS/GRAPH: Graph Template Language User's Guide*.

The following code creates the STATGRAPH template MyGraphs.Regplot, which creates the following graph.

```
proc template;
define statgraph mygraphs.regplot;
begingraph;
  entrytitle "Regression Plot";
  layout overlay;
 modelband "mean";
    scatterplot x=height y=weight;
 regressionplot x=height y=weight / clm="mean";
  endlayout;
endgraph;
end;
run;

ods listing style=analysis;
ods graphics / reset imagename="reg" width=500px;

proc sgrender data=sashelp.class template=mygraphs.regplot;
run;
```

Display 7.4 Graph Created with a STATGRAPH Template

The following display shows a scatter plot with an overlaid regression line and confidence limits of the mean for the HEIGHT and WEIGHT variables of a data set.



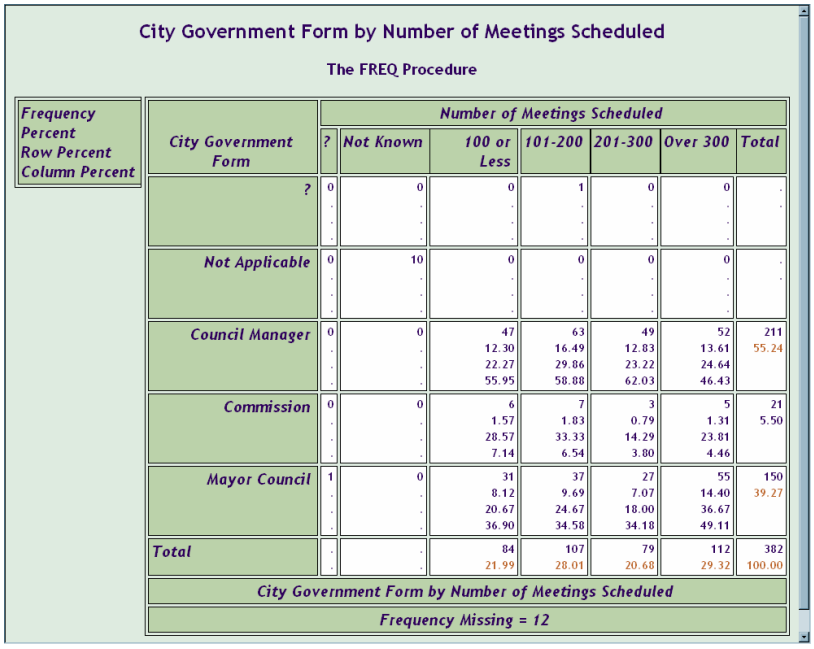
Modify a Crosstabulation Table

The TEMPLATE procedure enables you to customize the appearance of crosstabulation (contingency) tables that are created with the FREQ procedure. By default, crosstabulation tables are formatted according to the CrossTabFreqs template that SAS provides. However, you can create a customized CrossTabFreqs table template by using the TEMPLATE procedure with the DEFINE CROSSTABS statement.

This output shows the use of a customized crosstabulation table template for the CrossTabFreqs table. The program used to create the modified crosstabulation table template does the following:

- modifies table regions

- customizes legend text
- modifies headers and footers
- modifies variable labels used in headers
- customizes styles for cellvalues



City Government Form by Number of Meetings Scheduled

The FREQ Procedure

Frequency Percent Row Percent Column Percent	City Government Form	Number of Meetings Scheduled						
		?	Not Known	100 or Less	101-200	201-300	Over 300	Total
	?	0 . . .	0 . . .	0 . . .	1 . . .	0 . . .	0 . . .	. .
	Not Applicable	0 . . .	10 . . .	0 . . .	0 . . .	0 . . .	0 . . .	. .
	Council Manager	0 . . .	0 . . .	47 12.30 22.27 55.95	63 16.49 29.86 58.88	49 12.83 23.22 62.03	52 13.61 24.64 46.43	211 55.24
	Commission	0 . . .	0 . . .	6 1.57 28.57 7.14	7 1.83 33.33 6.54	3 0.79 14.29 3.80	5 1.31 23.81 4.46	21 5.50
	Mayor Council	1 . . .	0 . . .	31 8.12 20.67 36.90	37 9.69 24.67 34.58	27 7.07 18.00 34.18	55 14.40 36.67 49.11	150 39.27
	Total	. .	.	84 21.99	107 28.01	79 20.68	112 29.32	382 100.00
	City Government Form by Number of Meetings Scheduled							
	Frequency Missing = 12							

The Backward Compatibility of ODS Templates

ODS templates are not binary compatible between SAS versions. However, with some templates, you can use a template created with an earlier version of SAS with a later version of SAS. The following table lists the ODS templates and whether they are forward or backward compatible between SAS versions.

Table 7.1 Compatibility of ODS Templates between SAS Versions

ODS Template	Backward Compatible	Forward Compatible
table	no	yes
crosstabs	no	yes
style	no	yes *
tagset	no	no
ODS Graphics	no	no

* Styles that use inheritance may not be compatible forwards or backwards. See "Inheritance Compatibility across Versions" on page 548 for more information.

If you would like to use a template created with a later version of SAS with an earlier version of SAS, you might be able to extract the template source and use it to compile the template in the earlier release.

Terminology: TEMPLATE Procedure

These terms frequently appear in discussions of PROC TEMPLATE:

aggregate storage location
: is a location on an operating system that can contain a group of distinct files. Different host operating systems call an aggregate grouping of files different names, such as a directory, a maclib, or a partitioned data set. The standard form for referencing an aggregate storage location from within SAS is fileref(name), where fileref is the entire aggregate and (name) is a specific file or member of that aggregate.

graph template
: describes the contents and structure of a single-cell or multi-cell graph.

item store
: is a member of a SAS library. An item store is a hierarchical file system that is implemented as a single physical file. An item store can contain directories and files (called items) similar to the file systems in the UNIX and Windows operating environments. An item store is referenced by a two-level name: a libref and the name of the item store in the SAS library that the libref references. For example, the SAS registry is stored in two items stores, SASUSER.REGISTRY and SASHELP.REGISTRY.

template store
: is an item store that stores templates that were created by the TEMPLATE procedure. Templates that SAS provides are in the item store SASHELP.TMPLMST. You can store templates that you create in any template store where you have write access.

  Note: A template store can contain multiple levels known as directories. When you specify a template store in the ODS PATH statement, however, you specify a two-level name that includes a libref and the name of a template store in the SAS library that the libref references. △

style (template)
: describes how to display the presentation aspects (color, font face, font size, and so on) of your SAS output. A style determines the overall appearance of the documents that use it. Each style consists of style elements.

style element
: is a collection of style attributes that apply to a particular part of the output. For example, a style element can contain instructions for the presentation of column headings or for the presentation of the data inside cells. Style elements can also specify default colors and fonts for output that uses the style. Each style attribute specifies a value for one aspect of the presentation. For example, the BACKGROUND= attribute specifies the color for the background of an HTML table, and the FONTSTYLE= attribute specifies whether to use a Roman, a slant, or an italic font.

table template
: describes how to display the output for a tabular output object. (Most ODS output is tabular.) A table template determines the order of table headers and footers, the order of columns, and the overall appearance of the output object that uses it. Each table template contains or references table elements.

table element
: is a collection of attributes that apply to a particular column, header, or footer. Typically, these attributes specify something about the data rather than about its

presentation. For example, FORMAT= specifies the SAS format to use in a column. However, some attributes describe presentation aspects of the data.

Note: You can also define table elements such as columns, headers, and footers outside of a table template. Any table template can then reference these table elements. For more information about defining columns, headers, and footers outside of the table template, see Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593. △

tagset
specifies instructions for creating a markup language for your SAS output. The resulting output contains embedded instructions in order to define layout and some content. Each tagset contains event templates and event attributes that control the generation of the output. SAS provides tagsets for a variety of markup languages. With the TEMPLATE procedure, you can modify any of these SAS tagsets, or you can create your own tagsets.

event
specifies the text that the markup destination produces when the specified event occurs. For example, the template of an event called ROW might specify to place the appropriate tags for starting a row at the beginning of an event and the appropriate tags for ending a row at the end of the event. SAS procedures that generate ODS output use a standard set of events, which you can customize with the TEMPLATE procedure.

PROC TEMPLATE Statements by Category

This table lists and describes the categories and statements used in the TEMPLATE procedure.

Task	Statements Category	Statements	Description
Navigate template stores and manage ODS templates	Template store	DELETE	Deletes the specified template
		LINK	Creates a link to an existing template
		LIST	Lists items in one or more template stores
		PATH	Specifies the locations to write to or read from when creating or using PROC TEMPLATE templates, and the order in which to search for them
		SOURCE	Writes the source code for the specified template
		TEST	Tests the most recently created template by binding it to the specified data set

Task	Statements Category	Statements	Description
Create or modify ODS style	Style	DEFINE STYLE	Creates a style for any destination that supports the STYLE= option
Create and modify ODS table, column, header, and footer templates	Tabular	EDIT	Edits an existing template
		DEFINE COLUMN	Creates a template for a column
		DEFINE FOOTER	Creates a template for a table footer
		DEFINE HEADER	Creates a template for a header
		DEFINE TABLE	Creates a template for a table
Create or modify markup language tagsets	Markup language tagsets	DEFINE TAGSET	Creates a template for a tagset
Create or modify a crosstabulation table template	Tabular	DEFINE CROSSTABS	Creates a template for a PROC FREQ crosstabulation table
Create or modify a graph template	Graphical	DEFINE STATGRAPH	Creates a template for a graph

Syntax: TEMPLATE Procedure

PROC TEMPLATE;
  DEFINE COLUMN *column-path* </ STORE=*libref.template-store*>;
    <*column-attribute-1*; <*...column-attribute-n*;>>
    statements
    END;
  DEFINE FOOTER *footer-path* </ STORE=*libref.template-store*>;
    <*footer-attribute-1*; <*...footer-attribute-n*;>>
    statements
    END;
  DEFINE HEADER *template-name* </ STORE=*libref.template-store*>;
    <*header-attribute-1*; <*...header-attribute-n*;>>
    statements
    END;
  DEFINE STYLE *style-path* </ STORE=*libref.template-store*>;
    <**PARENT=***style-path*;>
    statements
    END;
  DEFINE TABLE *table-path* </ STORE=*libref.template-store*>;

<table-attribute-1; <*...table-attribute-n*;>>
statements
END;

DEFINE TAGSET *tagset-path* </ STORE=*libref.template-store*>;
DEFINE EVENT *event-name*;
<*event-attribute-1*; <*...event-attribute-n*;>>
statements
END;

DEFINE CROSSTABS *table-path* </ STORE=*libref.template-store*>;
statements
END;

DEFINE STATGRAPH *graph-path* </ STORE=*libref.template-store*>;
statements
END;

DELETE *template-path* </ STORE=*libref.template-store* >;

EDIT *template-path-1* <AS *template-path-2*> </ STORE=*libref.template-store* > ;
statements-and-attributes
END;

LINK *template-path-1* **TO***template-path-2* </ *option(s)*>;

LIST <*starting-path*></ *option(s)*>;

PATH *location(s)*;

SOURCE *template-path* </ option(s)>;

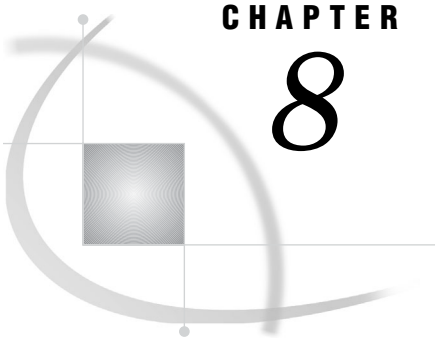
TEST DATA=*data-set* </ STORE=*libref.template-store*>;

Task	Statement
Create a column template	“DEFINE COLUMN Statement” on page 599
Create a crosstabulation template	“DEFINE CROSSTABS Statement” on page 433
Create a footer template	“DEFINE FOOTER Statement” on page 625
Create a header template	“DEFINE HEADER Statement” on page 626
Create a graph template	Chapter 10, “TEMPLATE Procedure: Creating ODS Graphics,” on page 483
Create a style	“DEFINE STYLE Statement” on page 490
Create a table template	“DEFINE TABLE Statement” on page 640
Create a tagset	“DEFINE TAGSET Statement” on page 796
Delete the specified template	“DELETE Statement” on page 409
Edit an existing template	“EDIT Statement” on page 597
Create a link to an existing template	“LINK Statement” on page 410
List items in one or more template stores	“LIST Statement” on page 411
Specify the locations to write to or read from when creating or using PROC TEMPLATE templates, and the order in which to search for them	“PATH Statement” on page 416

Task	Statement
Write the source code for the specified template to the SAS log	"SOURCE Statement" on page 417
Test the most recently created template by binding it to the specified data set	"TEST Statement" on page 422

Where to Go from Here

- *Creating statistical graphics with ODS:* For reference information about the Graph Template Language, see *SAS/GRAPH: Graph Template Language Reference*.
- *Creating statistical graphics with ODS:* For usage information about PROC TEMPLATE and the Graph Template Language, see *SAS/GRAPH: Graph Template Language User's Guide*.
- *Managing the various templates stored in template stores*: For reference information about the PROC TEMPLATE statements that help you manage and navigate around the many ODS templates, see Chapter 8, "TEMPLATE Procedure: Managing Template Stores," on page 407.
- *Modifying an existing style or creating your own style* : For reference information about the style definition statements in PROC TEMPLATE, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.
- *Creating and modifying ODS tabular output*: For reference information about the tabular template statements in PROC TEMPLATE, see Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593.
- *Modifying markup language tagsets that SAS provides or creating your own tagsets*: For reference information about the markup language tagset statements in PROC TEMPLATE, see Chapter 13, "TEMPLATE Procedure: Creating Markup Language Tagsets," on page 795.



CHAPTER

8

TEMPLATE Procedure: Managing Template Stores

Overview: Template Stores

What Is a Template Store?

A template store is an item store which stores items that were created by the TEMPLATE procedure. Items that SAS provides are in the item store SASHELP.TMPLMST. You can store items that you create in any template store where you have Write access.

Note: A template store can contain multiple levels known as directories. When you specify a template store in the ODS PATH statement, however, you specify a two-level name that includes a libref and the name of a template store in the SAS library that the libref references. △

Why Use the TEMPLATE Procedure to Manage Template Stores?

You can use the TEMPLATE procedure to manage and navigate the template stores that store the items that SAS supplies or that you create. The TEMPLATE procedure enables you to perform the following management tasks for the template stores:

- delete column templates, header templates, footer templates, styles, table templates, or tagsets
- list items in one or more template stores
- view the source code of column templates, header templates, footer templates, styles, table templates, or tagsets
- test the most recently created item

You can navigate around the template stores by doing the following:

- create links to existing items
- specify which locations to write to or read from when you create or use PROC TEMPLATE items, and specify the order in which to search for them

Terminology

For definitions of terms used in this section, see "Terminology: TEMPLATE Procedure" on page 402.

Template Store Syntax: TEMPLATE Procedure

PROC TEMPLATE;

DELETE *item-path*< / STORE=*libref.template-store*>;

LINK *item-path-1* **TO** *item-path-2* <*/option(s)*>;

LIST <*starting-path*></ *option(s)*>;

PATH *location(s)*;

SOURCE *item-path* <*/option(s)*><STORE=*libref.template-store*>;

TEST DATA=*data-set*< / STORE=*libref.template-store*>;

Task	Statement
Delete the specified item	"DELETE Statement" on page 409
Create a link to an existing item	"LINK Statement" on page 410
List items in one or more template stores	"LIST Statement" on page 411
Specify which locations to write to or read from when you create or use PROC TEMPLATE items, and specify the order in which to search for them	"PATH Statement" on page 416

Task	Statement
Write the source code for the specified item to the SAS log	"SOURCE Statement" on page 417
Test the most recently created item by binding it to the specified data set	"TEST Statement" on page 422

PROC TEMPLATE Statement

PROC TEMPLATE;

DELETE Statement

Deletes the specified item.

DELETE *item-path*;

Required Arguments

item-path
specifies an item to delete. An *item-path* consists of one or more names, separated by periods. Each name represents a directory in a template store. (A template store is a type of SAS file.) If the same item exists in multiple template stores, PROC TEMPLATE deletes the item from the first template store in the current path where you have Write access.

CAUTION:
Deleting a directory in a template store deletes all subdirectories and items in the directory. If the path that you specify is a directory rather than an item, PROC TEMPLATE deletes all the directories and all the items in that directory. △

LINK Statement

Creates a link to an existing item.

LINK *item-path-1* **TO** *item-path-2* </ *option(s)*>;

Creating a link to an item has the same effect as creating a new item that inherits its characteristics from another item (see the discussion of PARENT="PARENT= Statement" on page 494 statement). However, using a link is more efficient than using inheritance, because linking does not actually create a new item.

Required Arguments

item-path-1
specifies the path of the item to create. PROC TEMPLATE creates the item in the first template store in the path that you can write to.

item-path-2
specifies the path of the item to link to. If the same item exists in multiple template stores, PROC TEMPLATE uses the one from the first template store in the current path that you can read.

Tip: PROC TEMPLATE does not confirm that *item-path-2* exists when it compiles the item.

Options

NOTES= *'text'*
specifies notes to store in the item.

Requirement: Enclose the text in quotation marks.

Tip: Notes of this type become part of the compiled item, which you can view with the SOURCE statement, whereas SAS comments do not.

STORE=*libref.template-store*
specifies the location where the link will be created.

Restriction: The STORE= option syntax does not become part of the compiled item.

Tip: The link always points to the first item with the same name that it finds in the ODS path.

LIST Statement

Lists the items in one or more template stores.

Featured in: Example 1 on page 424

LIST <*starting-path*></ *option(s)*>;

Options

starting-path
specifies a level within each template store where PROC TEMPLATE starts listing items. For example, if *starting-path* is **`base.univariate`**, PROC TEMPLATE lists only **`base.univariate`** and the items within it and within all the levels that it contains.

Default: If you omit a *starting-path*, then the LIST statement lists all items in all template stores unless the ODS PATH statement confines the search to the specified template stores.

Restriction: This option must precede the forward slash (/) in the LIST statement.

SORT=*statistic <sorting-order>*
sorts the list of items by the specified statistic in the specified sorting order.

statistic
is one of the following:

CREATED
is the date that the item was created.

NOTES
is the content of any NOTES statement in the PROC TEMPLATE step that created the item.

Alias: LABEL

LINK
is the name of the item that the current item links to (see "LINK Statement" on page 410).

PATH
is the path to the current item in the template store. (The path does not include the name of the template store).

SIZE
is the size of the item.

TYPE
is the type of the item: COLUMN, FOOTER, HEADER, STYLE, TABLE, or LINK. If the item is simply a level in the item store, its type is DIR.

Default: PATH

sorting-order
specifies whether SORT= sorts from low values to high values or from high values to low values.

ASCENDING
sorts from low values to high values.

Alias: A

DESCENDING
sorts from high values to low values.

Alias: D

Default: ASCENDING

STATS=ALL | *(statistic-1 <, … statistic-n>)*
specifies the information to include in the list of items.

ALL
includes all available information.

(statistic-1 <, … statistic-n>)
includes the specified information. *statistic* is one or more of the following:

CREATED
is the date that the item was created.

NOTES
is the content of any NOTES statement in the PROC TEMPLATE step that created the item.

Alias: LABEL

LINK
is the name of the item that the current item links to (see "LINK Statement" on page 410).

SIZE
is the size of the item.

Default: Whether or not you specify STATS=, the list of items always includes an observation number, the path to the item, and its type.

STORE=*libref.template-store*
specifies the template store to process.

Default: All template stores in the current template path (see "PATH Statement" on page 416).

WHERE=*where-expression*
selects, for listing, items that meet a particular condition. For example, the following statement lists items that contain the word "Default" in the path to the current template:

```
list / where=(path ? 'Default');
```

where-expression
is an arithmetic or logical expression that consists of a sequence of operators and operands.

where-expression has this form:

(*subsetting-variable* <*comparison-operator where-expression-n*>)

subsetting-variable
a special kind of WHERE expression operand used by the SOURCE statement to help you find common values in items. Subsetting variables are one or more of the following:

PATH | _PATH_
is the fully qualified path of a template.

Alias: NAME | _NAME_

Alias: TEMPLATE | _TEMPLATE_

Example: This SOURCE statement displays the code for all items that contain the word "Default" in the name of the current template:

```
source / where=(path ? 'Default');
run;
```

TYPE | _TYPE_
is the type of the item. TYPE is one of the following:

COLUMN
specifies that the template is a column in a table.

FOOTER
specifies that the template is a footer in a table.

HEADER
specifies that the template is a header in a table.

LINK
specifies that the template is a link or URL.

STYLE
specifies that the definition is a style.

TABLE
specifies that the definition is a table template .

TAGSET
specifies that the definition is a tagset.

Example: This SOURCE statement displays the source code for all tagsets that have the word "Default" in the path :

```
source / where=(lowcase(type) = 'tagset' && _path_ ? 'Default');
```

The LOWCASE function converts all letters in an argument to lowercase.

NOTES
is the content of any NOTES statement in the PROC TEMPLATE step that created the item. The contents is displayed in the LABEL field.

Alias: LABEL

Example: This SOURCE statement displays the source code for all items where the label contains the words "common matrix" and the item is a link:

```
source / where=(lowcase(label) ? 'common matrix' && _type_ = 'Link');
run;
```

The LOWCASE function converts all letters in an argument to lowercase.

SIZE
is the size of the item in bytes.

Example: This SOURCE statement displays the source code for all items that are larger than 70000 bytes:

```
source / where=(size > 70000);
run;
```

CREATED
is the date the item was created.

Example: This SOURCE statement displays the source code for all of the items that were created today in all of the template stores in the current template path :

```
source / where=(datepart(created) = today());
```

The DATEPART function extracts the date from a SAS datetime value.

CDATE | _CDATE_
is the creation date of the item.

Example: This SOURCE statement displays the source code for all of the items with a creation date of 16JUL2004:

```
source / where=(_cdate_ = '16JUL2004'd);
run;
```

CDATETIME | _CDATETIME_
is the creation datetime of the item.

Example: This SOURCE statement displays the source code for all items with a creation SAS datetime of May 1, 2003 at 9:30:

```
source / where=(_cdatetime_ = '01may04:9:30:00'dt);
run;
```

CTIME | _CTIME_
is the creation time of the item.

Example: This SOURCE statement displays the source code of all items with a creation time of 9:25:19 PM:

```
source / where=(_ctime_ = '9:25:19pm't);
run;
```

MDATE | _MDATE_
is the modification date of the item.

Example: This SOURCE statement displays the source code of all items with a modification date of 16JUL2004:

```
source / where=(_mdate_ = '16JUL2004'd);
run;
```

MDATETIME | _MDATETIME_
is the modification datetime of the item.

Example: This SOURCE statement displays the source code of all items with a modification SAS datetime of May 1, 2003 at 9:30:

```
source / where=(_mdatetime_ = '01may04:9:30:00'dt);
run;
```

MODIFIED
is the date the item was modified.

Example: This SOURCE statement displays the source code of all items that were modified today in all of the template stores in the current template path :

```
source / where=(datepart(modified) = today());
```

The DATEPART function extracts the date from a SAS datetime value.

MTIME |_MTIME_
is the modification time of the item.

Example: This SOURCE statement displays the source code of all items with a modification time of 9:25:19 PM:

```
source / where=(_mtime_ = '9:25:19pm't);
run;
```

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 8.1 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

See also: For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

Featured in: Example 2 on page 425

PATH Statement

Specifies locations to write to or read from when you create or use PROC TEMPLATE templates or definitions, and specifies the order in which to search for them. This statement overrides the ODS PATH statement for the duration of the PROC TEMPLATE step.

Featured in: Example 1 on page 424 and Example 3 on page 426

PATH <(APPEND) | (PREPEND) | (REMOVE) > *location(s)*;

PATH *path-argument*;

Required Arguments

location(s)
specifies one or more locations to write to or read from when creating or using PROC TEMPLATE items and the order in which to search for them. ODS searches the locations in the order that they appear on the statement. It uses the first definition that it finds that has the appropriate access mode (Read, Write, or Update) set.

Each *location* has this form:

<*libref.*>*item-store* <(READ | UPDATE | WRITE)>

<*libref.*>*item-store*
identifies an item store to read from, to write to, or to update. If an item store does not already exist, then the PATH statement creates it.

(READ | UPDATE | WRITE)
specifies the access mode for the item. An access mode is one of the following:

READ
provides read-only access.

WRITE
provides Write access (always creating a new template store) as well as Read access.

UPDATE
provides Update access (creating a new template store only if the specified one does not exist) as well as Read access.

Default: READ

Default: The general default path is as follows:

SASUSER.TEMPLAT (UPDATE))
SASHELP.TMPLMST (READ)

If you have specified the RSASUSER SAS system option, then the default path is as follows:

WORK.TEMPLAT(UPDATE)
SASUSER.TEMPLAT (READ)
SASHELP.TMPLMST (READ)

Note: SAS stores all the items that it provides in SASHELP.TMPLMST. △

Note: See the RSASUSER SAS system option in *SAS Language Reference: Dictionary* for more information. △

Tip: If you want to be able to ignore all the items that you create, then keep them in their own item stores so that you can leave them out of the list of item stores that ODS searches.

path-argument
sets or displays the ODS path.
path-argument is one of the following:

RESET
sets the ODS path to the default settings SASUSER.TEMPLAT (UPDATE) and SASHELP.TMPLMST (READ).

SHOW
displays the current ODS path.

VERIFY
sets the ODS path to include only templates supplied by SAS. Specifying VERIFY is the same as specifying ODS PATH SASHELP.TMPLMST (READ).

Options

(APPEND | PREPEND | REMOVE)
adds one or more locations to a path, or removes one or more locations from a path.

APPEND
adds one or more locations to the end of a path. When you append a location to a path, all duplicate instances (with the same name and same permissions) of that item store are removed from the path. Only the last item store with the same name and permissions is kept.

PREPEND
adds one or more locations to the beginning of a path. When you prepend a location to a path, all duplicate instances (with the same name and same permissions) of that item store are removed from the path. Only the first item store with the same name and permissions is kept.

REMOVE
removes one or more locations from a path.

Default: If you omit an APPEND, PREPEND, or REMOVE option, then the PATH statement overwrites the complete path.

SOURCE Statement

Writes the source code for the template specified to the SAS log.

Featured in: Example 3 on page 426

SOURCE *item-path* </ *option(s)*>;

Required Arguments

item-path
specifies the path of the item that you want to write to the SAS log. If the same item exists in multiple template stores, PROC TEMPLATE uses the one from the first template store that you can read in the current path.

Tip: PROC TEMPLATE stores items in compiled form. The SOURCE statement actually decompiles the item. Because SAS comments are not compiled, comments that are in the source code do not appear when you decompile the item. If you want to annotate the item, use the NOTES statement inside the item or the block of editing instructions, or use the NOTES= option in the LINK statement. These notes do become part of the compiled item. (See "NOTES Statement" on page 660 and the discussion of the NOTES= option on page 410. You can also specify notes as quoted strings in the DYNAMIC, MVAR, NMVAR, REPLACE, and STYLE statements.)

Options

FILE= *'file-specification'* | *fileref*
specifies a file to write the item to.

'file-specification'
is the name of an external file to write to.

Requirement: The *external-file* that you specify must be enclosed in quotation marks.

fileref
is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

Default: If you omit a filename where you want the source code written, then the SOURCE statement writes the source code to the SAS log.

See: "Statements" in *SAS Language Reference: Dictionary* for information about the FILENAME statement.

NOFOLLOW
specifies that the program not resolve links in the PARENT= statement, which specifies the item that the current item inherits from. For information about the PARENT= statement, see the PARENT="PARENT= Statement" on page 494 statement in the styles attribute section.

STORE= *libref.template-store*
specifies the template store where the item is located.

Interaction: In most cases, the STORE= option is added to the definition statement when PROC TEMPLATE displays the source code. However, if the template store specified in the STORE= option is in the ODS path with only read permission, then PROC TEMPLATE does not include the STORE= option in the source code that it displays. There will be no STORE= option, which means that if you run the code, then the item that it creates will go to the first template store for which you have Update permission in the ODS path.

WHERE=(*where-expression*)
selects items that meet a particular condition. For example, the following statement displays the source code for items that contain the word "Default" in the path to the current template:

```
source / where=(path ? 'Default');
```

where-expression
is an arithmetic or logical expression that consists of a sequence of operators and operands.

where-expression has this form:

(*subsetting-variable* <*comparison-operator where-expression-n*>)

subsetting-variable
a special kind of WHERE expression operand used by the SOURCE statement to help you find common values in items. Subsetting variables are one or more of the following:

PATH | _PATH_
is the fully qualified path of a template.

Alias: NAME | _NAME_

Alias: TEMPLATE | _TEMPLATE_

Example: This SOURCE statement displays the code for all items that contain the word "Default" in the name of the current template:

```
source / where=(path ? 'Default');
run;
```

TYPE | _TYPE_
is the type of the item. TYPE is one of the following:

COLUMN
specifies that the template is a column in a table.

FOOTER
specifies that the template is a footer in a table.

HEADER
specifies that the template is a header in a table.

LINK
specifies that the template is a link or URL.

STYLE
specifies that the definition is a style.

TABLE
specifies that the definition is a table template .

TAGSET
specifies that the definition is a tagset.

Example: This SOURCE statement displays the source code for all tagsets that have the word "Default" in the path :

```
source / where=(lowcase(type) = 'tagset' && _path_ ? 'Default');
```

The LOWCASE function converts all letters in an argument to lowercase.

NOTES

is the content of any NOTES statement in the PROC TEMPLATE step that created the item. The contents is displayed in the LABEL field.

Alias: LABEL

Example: This SOURCE statement displays the source code for all items where the label contains the words “common matrix” and the item is a link:

```
source / where=(lowcase(label) ? 'common matrix' && _type_ = 'Link');
run;
```

The LOWCASE function converts all letters in an argument to lowercase.

SIZE
is the size of the item in bytes.

Example: This SOURCE statement displays the source code for all items that are larger than 70000 bytes:

```
source / where=(size > 70000);
run;
```

CREATED
is the date the item was created.

Example: This SOURCE statement displays the source code for all of the items that were created today in all of the template stores in the current template path :

```
source / where=(datepart(created) = today());
```

The DATEPART function extracts the date from a SAS datetime value.

CDATE | _CDATE_
is the creation date of the item.

Example: This SOURCE statement displays the source code for all of the items with a creation date of 16JUL2004:

```
source / where=(_cdate_ = '16JUL2004'd);
run;
```

CDATETIME | _CDATETIME_
is the creation datetime of the item.

Example: This SOURCE statement displays the source code for all items with a creation SAS datetime of May 1, 2003 at 9:30:

```
source / where=(_cdatetime_ = '01may04:9:30:00'dt);
run;
```

CTIME | _CTIME_
is the creation time of the item.

Example: This SOURCE statement displays the source code of all items with a creation time of 9:25:19 PM:

```
source / where=(_ctime_ = '9:25:19pm't);
run;
```

MDATE | _MDATE_
is the modification date of the item.

Example: This SOURCE statement displays the source code of all items with a modification date of 16JUL2004:

```
source / where=(_mdate_ = '16JUL2004'd);
run;
```

MDATETIME | _MDATETIME_
is the modification datetime of the item.

Example: This SOURCE statement displays the source code of all items with a modification SAS datetime of May 1, 2003 at 9:30:

```
source / where=(_mdatetime_ = '01may04:9:30:00'dt);
run;
```

MODIFIED
is the date the item was modified.

Example: This SOURCE statement displays the source code of all items that were modified today in all of the template stores in the current template path :

```
source / where=(datepart(modified) = today());
```

The DATEPART function extracts the date from a SAS datetime value.

MTIME |_MTIME_
is the modification time of the item.

Example: This SOURCE statement displays the source code of all items with a modification time of 9:25:19 PM:

```
source / where=(_mtime_ = '9:25:19pm't);
run;
```

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 8.2 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

See also: For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

TEST Statement

Tests the most recently created item by binding it to the specified data set.

TEST DATA= *data-set* </ STORE=*libref.template-store*>;

Required Arguments

DATA=*data-set*
specifies the SAS data set to bind to the most recently created item. ODS sends this output object to all open ODS destinations.

Options

STORE=*libref.template-store*
specifies the template store where the item is located.

Requirement: If you specify this option, then the template store that you specify must match the template store in the DEFINE statement that created the item.

Concepts: Template Stores and the TEMPLATE Procedure

The Contents of Templates That SAS Supplies

SAS provides templates for these items:

- tables
- crosstabulation tables
- SAS statistical graphics
- styles
- tagsets

To view the contents of a template, use the SAS windowing environment, the SAS window command ODSTEMPLATES, or the TEMPLATE procedure.

- *SAS Windowing Environment*
  1. From the SAS Explorer, select **View ▶ Results**.
  2. In the Results window, select the **`Results`** folder. Right–click to open the Templates window.
  3. To view the definitions or templates that SAS supplies, click the plus sign that is next to the SASHELP.TMPLMST item store.
  4. Click the plus sign that is next to an icon to view the contents of that template store or directory in a template store. If there is no plus sign next to the icon, double-click the icon to view the contents of that directory.

□ *SAS Windowing Command*

1 To view the Templates window, submit this command in the command bar:

```
odstemplates
```

This display shows the Templates window that contains the item stores **`Sasuser.Templat`** and **`Sashelp.Tmplmst`**.

2 When you double-click an item store, such as **`Sashelp.Tmplmst`**, that item store expands to list the directories where ODS templates are stored. The templates that SAS provides are in the item store **`Sashelp.Tmplmst`**.

□ *TEMPLATE Procedure*

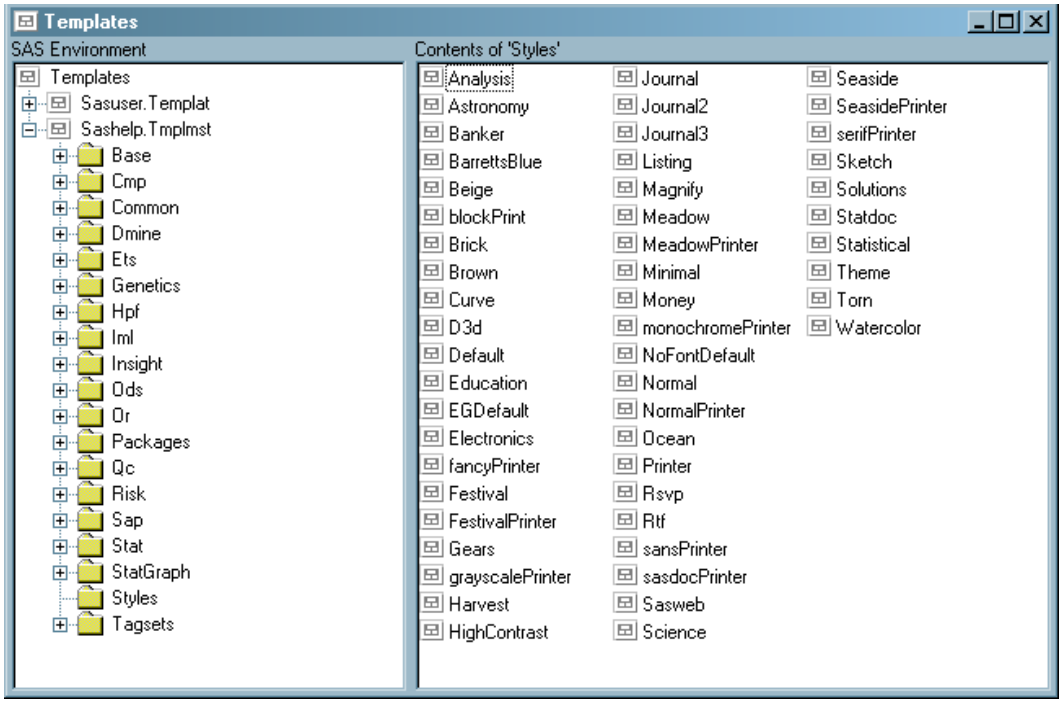
The SOURCE statement writes the source code for the specified template to the SAS log. For example, if you want to view the source for all the objects in Base SAS, submit this code:

```
proc template;
source base;
run;
```

Note: For more information, see "SOURCE Statement" on page 417. △

Display 8.1 Templates That SAS Supplies

From the Templates window, you can see the definitions and templates that SAS supplies or that you created. This figure shows the styles that SAS supplies.



Examples: Managing Template Stores Using the TEMPLATE Procedure

Example 1: Listing Templates in a Template Store

PROC TEMPLATE features:

PATH statement

LIST statement:

starting-path option

SORT= option

Program Description

This example lists the items for the Base.Univariate directory in the item store SASHELP.TMPLMST.

Program

Set the SAS system options. The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Specify which locations to search for items that were created by PROC TEMPLATE. The PATH statement specifies to search for templates and definitions that were created by PROC TEMPLATE in the SASHELP.TMPLMST item store.

```
proc template;
path sashelp.tmplmst;
```

List in descending order the items that are stored within a specified level of the template store. The LIST statement lists the templates and definitions in one or more template stores. The starting path **`base.univariate`** specifies the level within the template store where PROC TEMPLATE is to start listing the items. The SORT= option sorts the list of items. The items are sorted in descending order.

```
list base.univariate / sort=path descending;
run;
```

Output 8.1 Listing of Base.Univariate Template Store

1

Listing of: SASHELP.TMPLMST
Path Filter is: Base.Univariate
Sort by: PATH/DESCENDING

Obs	Path	Type
1	Base.Univariate.Wins	Table
2	Base.Univariate.Trim	Table
3	Base.Univariate.Robustscale	Table
4	Base.Univariate.Quantiles	Table
5	Base.Univariate.PValue	Column
6	Base.Univariate.Normal	Table
7	Base.Univariate.Moments	Link
8	Base.Univariate.Modes	Table
9	Base.Univariate.Missings	Table
10	Base.Univariate.Measures	Table
11	Base.Univariate.Location	Table
12	Base.Univariate.LocCount	Table
13	Base.Univariate.Frequency	Table
14	Base.Univariate.FitQuant	Table
15	Base.Univariate.FitParms	Table
16	Base.Univariate.FitGood	Table
17	Base.Univariate.ExtVal	Table
18	Base.Univariate.ExtObs	Table
19	Base.Univariate.ConfLimits	Table
20	Base.Univariate.Bins	Table
21	Base.Univariate.BinPercents	Table
22	Base.Univariate	Dir

Example 2: Using a WHERE Expression to Select Items in a Template Store

PROC TEMPLATE features:

PATH statement

LIST statement:

where-expression option
starting-path option
SORT= option

Program Description

This example uses a WHERE expression to select, for listing, fitted distribution table templates in the Base.Univariate directory.

Program Program

Set the SAS system options. The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Specify which locations to search for items that were created by PROC TEMPLATE. The PATH statement specifies to search for items that were created by PROC TEMPLATE in the SASHELP.TMPLMST item store.

```
proc template;
path sashelp.tmplmst;
```

List, in descending order, the items with the word "fit" in their path name. The LIST statement lists the items in one or more template stores. The starting path **base.univariate** specifies the level within the template store where PROC TEMPLATE is to start listing the items. The WHERE expression finds items in the template store that have the word "fit" in their path name. The LOWCASE function converts all letters in an argument to lowercase. The SORT= option sorts the list of items. The items are sorted in descending order.

```
list base.univariate / sort=path descending
where=(lowcase(path) ? 'fit');
run;
```

Output 8.2 Listing of Fitted Distribution Templates In the Base.Univariate Template Store

1

Listing of: SASHELP.TMPLMST
Path Filter is: Base.Univariate
Sort by: PATH/DESCENDING

Obs	Path	Type
1	Base.Univariate.FitQuant	Table
2	Base.Univariate.FitParms	Table
3	Base.Univariate.FitGood	Table

Example 3: Viewing the Source of a Template

PROC TEMPLATE features:

PATH statement

SOURCE statement

Program Description

This example displays the source code for the Xhtml tagset that SAS provides.

Program

Specify which locations to search for items that were created by PROC TEMPLATE. The PATH statement specifies to search for items that were created by PROC TEMPLATE in the SASHELP.TMPLMST item store.

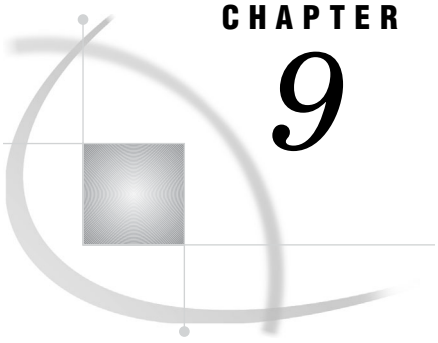
```
proc template;
path sashelp.tmplmst;
```

Write the source code of the specified item. The SOURCE statement writes the source code for the tagset Xhtml that SAS provides. The source code is written to the SAS log.

```
source Tagsets.Xhtml;
run;
```

Output 8.3 Source Code of the Template Tagset.Xhtml That Is Written to the SAS Log

```
  proc template;
 path sashelp.tmplmst;

    source Tagsets.Xhtml;
define tagset Tagsets.Xhtml;
   notes "XHTML 1.0";

   define event doc;
      start:
         set $empty_tag_suffix " /";
         set $doctype
               "";
         set $framedoctype
               "";
         put $doctype NL;
         put "" NL;

      finish:
         put "" NL;
   end;
   split = "";
   parent = tagsets.html4;
end;
NOTE: Path 'Tagsets.Xhtml' is in: SASHELP.TMPLMST.
  run;
NOTE: PROCEDURE TEMPLATE used (Total process time):
      real time           2.17 seconds
      cpu time            0.17 seconds
```



CHAPTER

9

TEMPLATE Procedure: Creating Crosstabulation Table Templates

Overview: ODS Crosstabulation Table Template Output

Using the TEMPLATE Procedure to Create a Customized Crosstabulation Table

The TEMPLATE procedure enables you to customize the appearance of crosstabulation (contingency) tables that are created with the FREQ procedure.

By default, crosstabulation tables are formatted according to the CrossTabFreqs template that SAS provides. However, you can create a customized CrossTabFreqs table template by using the TEMPLATE procedure with the statements in the following table.

Table 9.1 PROC TEMPLATE Statements

Task	Statement
Create a crosstabulation table template	DEFINE CROSSTABS
Define a value that appears in the crosstabulation cells	DEFINE CELLVALUE
Specify the order in which the cellvalues are stacked in the cells	CELLVALUE
Create a template for a footer	DEFINE FOOTER
Create a template for a header	DEFINE HEADER
End a crosstabulation table template	END

What Can You Do with a Crosstabulation Template?

The CrossTabFreqs crosstabulation template describes how to display PROC FREQ's crosstabulation table. You can create a customized CrossTabFreqs crosstabulation template to do the following:

- use custom formats for cellvalues
- specify a style for each value in a cell
- change the stacking order of values in a cell
- change and style headers and footers
- use variable labels in headers and footers
- style table regions independently
- change or remove the legend

The following display shows a crosstabulation table that has been created with the default crosstabulation table template:

Display 9.1 Crosstabulation Table Created with Default Crosstabulation Table Template

The FREQ Procedure

Frequency Deviation Tot Pct Percent Row Pct Col Pct Cumulative Col%	Table 1 of Treatment by Discomfort Controlling for Gender=F			
	Treatment(Treatment Regimen)	Discomfort(Amount of Discomfort Experienced)		
		P	PF	Total
	A	1 -1.667 1.67 3.33 10.00 12.50 12.50	9 1.6667 15.00 30.00 90.00 40.91 40.91	10 16.67 33.33 33.33
	B	1 -1.667 1.67 3.33 10.00 12.50 25.00	9 1.6667 15.00 30.00 90.00 40.91 81.82	10 16.67 33.33 66.67
	P	6 3.3333 10.00 20.00 60.00 75.00 100.00	4 -3.333 6.67 13.33 40.00 18.18 100.00	10 16.67 33.33 100.00
	Total	8 13.33 26.67	22 36.67 73.33	30 50.00 100.00

The following display shows PROC FREQ output that has been created with a modified CrossTabFreqs template:

Display 9.2 Crosstabulation Table Created with Modified Crosstabulation Table Template

The FREQ Procedure

Frequency
Deviation
Percent
Row Percent
Column Percent
Cum. Col. Percent
Total Percent

Treatment by Discomfort (Table 1)			
Gender of Patient=F			
Treatment Regimen	Amount of Discomfort Experienced		
	P	PF	Total
A	1	9	10
	-1.667	1.6667	
	3.33	30.00	33.33
	10.00	90.00	
	12.50	40.91	
	12.50	40.91	33.33
	1.67	15.00	16.67
B	1	9	10
	-1.667	1.6667	
	3.33	30.00	33.33
	10.00	90.00	
	12.50	40.91	
	25.00	81.82	66.67
	1.67	15.00	16.67
P	6	4	10
	3.3333	-3.333	
	20.00	13.33	33.33
	60.00	40.00	
	75.00	18.18	
	100.00	100.00	100.00
	10.00	6.67	16.67
Total	8	22	***30***
	26.67	73.33	***100.00***
	13.33	36.67	***50.00***

The following are some of the customizations that were made to the crosstabulation table above:

- The legend text has been italicized and made smaller than the rest of the header text.
- The header text now uses the variable label “Gender of Patient” instead of the variable name.
- Row variable labels and column variable labels are used now instead of row variable names and column variable names.
- The background color of the non-summary rows alternates.
- The values in the grand total cell are now bold and italic.
- The Deviation cellvalues are now red when the deviation exceeds abs (2.0).
- The TotalPercent cellvalue has been moved from the middle of the other cellvalues to the bottom of the cellvalues.

Crosstabulation Table Syntax: TEMPLATE Procedure

PROC TEMPLATE
  DEFINE CROSSTABS *table-path* </ STORE=*libref.template-store*>;
    <*table-attribute-1*; <*…table-attribute-n*>;>
    CELLVALUE *cellvalues*;
    DEFINE CELLVALUE *cellvalue*;
      statements-and-attributes;
    END;
    DEFINE HEADER *header-name*;
      statements-and-attributes;
    END;
    DEFINE FOOTER *footer-name*;
      statements-and-attributes;
    END;
    DYNAMIC *variable-1 <'text-1'> <…variable-n <'text-n'>>*;
    FOOTER *footer-name(s)*;
    HEADER *header-name(s)*;
    NOTES *text*;
  END;

Table 9.2 PROC TEMPLATE Statements That Add Different Features to Crosstabulation SAS Output

Task	Statement
Create a crosstabulation table template	DEFINE CROSSTABS
Define a value that appears in the crosstabulation cells	DEFINE CELLVALUE
Specify the order in which the cellvalues are stacked in the cells	CELLVALUE
Create a template for a footer	DEFINE FOOTER
Create a template for a header	DEFINE HEADER
End a crosstabulation table template	END

DEFINE CROSSTABS Statement

Creates a crosstabulation table template.

Featured in: Example 1 on page 460

DEFINE CROSSTABS *table-path* </ STORE=*libref.template-store*>;

<table-attribute-1; *<…table-attribute-n*>;>
CELLVALUE *cellvalues*;
DEFINE CELLVALUE *cellvalue*;
statements-and-attributes;
END;
DEFINE HEADER *header-name*;
statements-and-attributes;
END;
DEFINE FOOTER *footer-name*;
statements-and-attributes;
END;
DYNAMIC *variable-1 <'text-1'> <…variable-n <'text-n'>>*;
FOOTER *footer-name(s)*;
HEADER *header-name(s)*;
NOTES *text*;
END;

Table 9.3 DEFINE CROSSTABS Statements

Task	Statement
Set one or more cell attributes	*crosstabs-attributes*
Specify the order in which the cellvalues are stacked in the cells	CELLVALUE
Define a value that appears in the crosstabulation table's cells	DEFINE CELLVALUE
Create a template for a crosstabulation table header	DEFINE HEADER
Create a template for a crosstabulation table footer	DEFINE FOOTER
Define a symbol that references a value that the data component supplies from the procedure	DYNAMIC
Declare a symbol as a footer in the table and specify the order of the footers	FOOTER
Declare a symbol as a header in the table and specify the order of the headers	HEADER
Provide information about the crosstabulation table	NOTES
End a template, or end the editing of a template	END

Required Arguments

table-path
specifies where to store the crosstabulation table template. A table-path consists of one or more names, separated by periods. Each name represents a directory in a template store, which is a type of SAS file. For more information on template stores,

see “Understanding Item Stores, Template Stores, and Directories” on page 31. PROC TEMPLATE writes the template to the first writable template store in the current path.

Requirement: Crosstabulation table templates must be named **`CrossTabFreqs`**.

Options

STORE= *template-store*
specifies the template store in which to store the crosstabulation template. If the template store does not exist, it is created.

Restriction: The STORE= option does not become part of the template.

Requirement: The STORE= option must be proceeded by the forward slash (/) symbol.

Restriction: If the template is nested inside another template, do not use the STORE= option for the nested template, because the nested template is stored where the original template is stored.

DEFINE CROSSTABS Attributes

This section lists all of the attributes that you can use in a crosstabulation template.

Table 9.4 Crosstabulation Attributes

Task	Statement
Specify a style element and any changes to its attributes to use for the cellvalues in the non-summary rows and columns	CELL_STYLE=
Specify the name of the header to use over the column variable value columns in the table	COLS_HEADER=
Specify the style element and any changes to its attributes to use for the cellvalues in the last row in the table	COL_TOTAL_STYLE=
Specify the style element and any changes to its attributes to use for the column variable values used as headers over the column variable value columns	COL_VAR_STYLE=
Specify the style element and any changes to its attributes to use for the cellvalues in the rightmost column of the last row in the table	GRAND_TOTAL_STYLE=
Specify a label for the table	LABEL=
Specify the style element and any changes to its attributes to use for the legend table that appears near the upper-left corner of the table	LEGEND_STYLE=
Specify the name of the header to use over the row variable values (leftmost) column in the table	ROWS_HEADER=

Task	Statement
Specify the style element and any changes to its attributes to use for the cellvalues in the cells that contain row totals	ROW_TOTAL_STYLE=
Specify the style element and any changes to its attributes to use for the row variable values in the leftmost column of the table	ROW_VAR_STYLE=
Specify a style element and any changes to its attributes to use for the table	STYLE=

CELL_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes you can use for the cellvalues in the non-summary rows and columns. This refers to the cellvalues that are not in the row totals column (see ROW_TOTAL_STYLE), the column totals row (see COL_TOTAL_STYLE), or the grand total cell (see GRAND_TOTAL_STYLE).

Default: Data

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

COLS_HEADER=*header-name*
specifies the name of the header to use over the column variable value columns in the table.

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

COL_TOTAL_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the cellvalues in the last row in the table.

Default: Data

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

COL_VAR_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the column variable values used as headers over the column variable value columns.

Default: Header

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

GRAND_TOTAL_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the cellvalues in the rightmost column of the last row in the table.

Default: Data

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

LABEL="*text*"
specifies a label for the table.

Default: "Frequency Counts and Percentages"

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

LEGEND_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the legend table that appears near the upper-left corner of the table.

Default: Header

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

ROWS_HEADER=*header-name*
specifies the name of the header to use over the row variable values (leftmost) column in the table.

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

ROW_TOTAL_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the cellvalues that contain row totals.

Default: Data

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

ROW_VAR_STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the row variable values in the leftmost column of the table.

Default: RowHeader

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
Specifies the style element and any changes to its attributes to use for the table.

style-element-name
is the name of the style element to use to display the table. The style element must be part of a style that is registered with the Output Delivery System. SAS provides some style. You can create customized styles with PROC TEMPLATE (see "DEFINE STYLE Statement" on page 490). By default, ODS produces different parts of ODS output with different elements. For example, by default, a table is produced with the style element Table. The Table style element that SAS is provides is uniquely designed to describe elements necessary to a table. However, you might have a user-defined style element at your site that would be appropriate to specify.

The style element provides the basis for displaying the table. Additional style attributes that you provide can modify the display.

style-element-name is either the name of a style element or a variable whose value is a style element.

See also: "Viewing the Contents of a Style" on page 538

See also: "Working with Styles" on page 538

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

style-attribute-specification
describes the style attribute to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

See also: "Style Attributes and Their Values" on page 498

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

Default: Table

See also: "Crosstabulation Table Regions and Corresponding Attributes" on page 459 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region

---

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Tip: A dynamic variable that is defined in a template is available to that template and all the templates that it contains.

DYNAMIC *dynamic-variable(s)* ;

Required Arguments

dynamic- variable(s)
is a variable that is defined by SAS in the crosstabulation template. After a dynamic variable has been defined, you can use it in the TEXT statement within a footer or header template.

FMISSING
is the number of missing values in the table.

Requirement: The FMISSING dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

NOTITLE
is set to 1 if the PROC FREQ's NOTITLE option was used, and it is set to 0 if the NOTITLE option was not used.

Requirement: The NOTITLE dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

SAMPLESIZE
is set to 0 if the table is empty. Otherwise, it is set to 1.

Requirement: The SAMPLESIZE dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

STRATNUM
is the current stratum number if the table has multiple strata. If the table has only one stratum, then the value is 0.

Requirement: The STRATNUM dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

Featured in: Example 1 on page 460

STRATAVARIABLENAMES
is a string that identifies the current stratum by the name of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined if the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before the dynamic variables can be used in an expression.

STRATAVARIABLELABELS
is a string that identifies the current stratum by the label of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined when the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before you can use the dynamic variables in an expression.

Featured in: Example 1 on page 460 and Example 2 on page 469

FOOTER Statement

Declares a symbol as a footer in the table and specifies the order of the footers.

FOOTER *footer-specification(s)*;

Required Arguments

footer-specification(s)

specifies a symbol defined by the DEFINE FOOTER statement within the same table template.

Default: If you omit a FOOTER statement, ODS creates a footer for each footer template (DEFINE FOOTER statement) and places the footers in the same order that the footer templates have in the table template.

See also: "DEFINE FOOTER Statement" on page 449

HEADER Statement

Declares a symbol as a header in the table and specifies the order of the headers.

HEADER *header-specification(s)*;

Required Arguments

header-specification(s)
specifies a symbol defined by the DEFINE HEADER statement within the same table template.

Default: If you omit a HEADER statement, then ODS makes a header for each header template (DEFINE HEADER statement) and places the headers in the same order that the header templates have in the table template.

NOTES Statement

Provides information about a template.

Tip: The NOTES statement becomes part of the compiled template, which you can view with the SOURCE statement. SAS comments do not become part of the compiled template.

NOTES '*text*';

Required Arguments

'text'
provides information about the template.

END Statement

Ends a template.

END;

DEFINE CELLVALUE Statement

Defines a value that appears in the crosstabulation cells.

Featured in: Example 1 on page 460

DEFINE CELLVALUE <*cellvalue*>;
<*cellvalue-attribute-1*;><*…cellvalue-attribute-n*>;
CELLSTYLE *expression-1* **AS** <*style-element-name*><[*style-attribute-specification(s)*]
><*…*, *expression-n* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>>;
DYNAMIC *variable-1*<'*text-1*'> <*… variable-n*<'*text-n*'>>;
NOTES '*text*';
END;

Table 9.5 DEFINE CELLVALUE Statements

Task	Statement
Set one or more cellvalue attributes	*cellvalue-attributes*
Specify the style element of the cells in the column according to the values of the variables	CELLSTYLE AS
Define a symbol that references a value that the data component supplies from the procedure	DYNAMIC
Provide information about the crosstabulation table	NOTES
End a cellvalue template	END

Options

cellvalue
specifies one of the possible values that PROC FREQ can produce for a crosstabulation table. For a cellvalue to appear in a cell, it must meet one of these requirements:

- specified in a DEFINE CELLVALUE statement
- included in the CELLVALUE statement
- not suppressed by one of the following options for the TABLES statement in PROC FREQ: NOFREQ, NOPERCENT, NOROW, NOCOL, or CUMCOL
- requested by one of the following options: EXPECTED, DEVIATION, CELLCHI2, or TOTPCT

To prevent a cellvalue from appearing in a table, you need to change only one of the preceding specifications.

cellvalue is one of the following:

Frequency
is the frequency count.

Expected
is the expected frequency of the cell.

Deviation
is the deviation of the cell frequency from the expected value.

CellChiSquare
is the cell's contribution to the total Pearson chi-square statistic.

TotalPercent
is the percentage of total frequency on **n**-way tables when **n>2**.

Percent
is the percentage of the table frequency.

RowPercent
is the percentage of the row frequency.

ColPercent
is the percentage of the column frequency.

CumColPercent
is the cumulative percentage of the column frequency.

DEFINE CELLVALUE Attribute Statements

This section lists all the attribute statements that you can use in a cellvalue template and the tasks that are associated with the statements. For all attributes that support a value of ON, these forms are equivalent:

```
ATTRIBUTE-NAME
ATTRIBUTE-NAME=ON
```

Table 9.6 DEFINE CELLVALUE Attribute Statements

Task	Statement
Specify which format to use for the cellvalue if both a crosstabulation template and a data component specify a format	DATA_FORMAT_OVERRIDE=
Specify the format for the cellvalue	FORMAT=
Override the width specified by the FORMAT= attribute	FORMAT_WIDTH=
Override the number of decimals specified by the FORMAT= attribute	FORMAT_NDEC=
Specify the text in the legend	HEADER=
For the Output destination, specify the label for the data set column corresponding to the cellvalue	LABEL=
Specify whether a cellvalue appears in the crosstabulation table	PRINT=

DATA_FORMAT_OVERRIDE=<ON | OFF>;
specifies which format to use if both a crosstabulation template and a data component specify a format for Frequency, Expected, and Deviation.

ON
selects the format specified in the data component.

OFF
selects the format specified in the crosstabulation template.

Default: OFF

Interaction: If you specify DATA_FORMAT_OVERRIDE=ON, and the FORMAT option is specified on the TABLES statement in PROC FREQ, then the data component will specify that format for the Frequency, Expected, and Deviation cellvalues.

FORMAT=*format_name <format-width<decimal-width >>*;
specifies the format for the column.

Default: If you omit the FORMAT= option, PROC TEMPLATE uses the format that the data component provides. If the data component does not provide a format, PROC TEMPLATE uses one of the following:

- BEST8. for integers
- 12.3 for floating-point values
- the length of the variable for character variables

Interaction: For Listing output, the width of the cells is governed by the format width. Cells are at least one character wider than the format width.

Range: The minimum cell width is 8, and the maximum width is 25.

FORMAT_WIDTH=*positive-integer*
overrides the width specified by the FORMAT= attribute statement.

See also: FORMAT= on page 444

FORMAT_NDEC=*positive-integer*
overrides the number of decimals specified by the FORMAT= attribute statement.

See also: FORMAT= on page 444

HEADER='*text*'
specifies the text in the legend.

Tip: For Listing output, only the first 15 characters of text are displayed.

LABEL='*text*'
for the Output destination, specifies the label for the data set column that corresponds to the cellvalue.

PRINT= ON | OFF
specifies whether the cellvalue appears in the crosstabulation table.

Both this attribute and the TABLES statement option for the cellvalue control the presence of the cellvalue in the table. For example, the expected cell frequency is present only when the EXPECTED option is used and the Expected cellvalue template has PRINT=ON specified.

CELLSTYLE AS Statement

Sets the style element of the cells in the column according to the values of the variables. Use this statement to set the presentation characteristics (such as foreground color, font face, and flyover) of individual cells in all destinations except the LISTING destinations.

Featured in: Example 1 on page 460

CELLSTYLE *expression-1* **AS** <*style-element-name*><[*style-attribute-specification(s)*]><…, *expression-n* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>>;

Required Arguments

expression
is an expression that is evaluated for each cell. If *expression* resolves to TRUE (a non-zero value), the style element that is specified is used for the current cell. If *expression* is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

expression has this form:

expression-1 <comparison-operator expression-n>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

1
is a fixed value that you can use to set a constant style element.

Example: These statements set the cellvalue Frequency background to gray:

```
define cellvalue Frequency;
    other--statements ...;
   cellstyle 1 as {backgroundcolor=gray};
end;
```

VAL
is the value of the current cell.

Example: The following statements change the foreground color of the cellvalue Percent depending on its magnitude:

```
define cellvalue Percent;
   other--statements ...;
   cellstyle _val_ > 75.00 as {color=red},
      _val_ > 50.00 as {color=orange},
      _val_ > 25.00 as {color=green};
end;
```

comparison-operator
compares a variable with a value or with another variable.

Table 9.7 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Tip: Using an expression of 1 as the last expression in the CELLSTYLE AS statement sets the style element for any cells that did not meet an earlier condition.

Options

Note: Neither *style-attribute-specification* nor *style-element-name* is required. However, you must use at least one of them. △

style-attribute-specification
describes a style attribute to set. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

For information on the style attributes that you can set in a column template, see "Style Attributes and Their Values" on page 498.

Default: If you do not specify any style attributes to modify, ODS uses the unmodified *style-element-name*.

style-element-name
is the name of the style element that displays the data in the column. The style element must be part of a style that is registered with the Output Delivery System. SAS provides some styles. You can create customized styles by using PROC TEMPLATE (see "DEFINE STYLE Statement" on page 490). By default, ODS displays different parts of ODS output with different style elements. For example, by default, the data in a column is displayed with the style element Data. The style elements that you would probably use with the CELLSTYLE AS statement in a column template are the following.

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataEmphasisFixed
- DataStrong
- DataStrongFixed

The style element provides the basis for displaying the column. Additional style attributes that you provide can modify the display.

Default: Data

See also: "Viewing the Contents of a Style" on page 538

See also: "Working with Styles" on page 538

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Tip: A dynamic variable that is defined in a template is available to that template and all the templates that it contains.

DYNAMIC *dynamic-variable(s)* ;

Required Arguments

dynamic- variable(s)
is a variable that is defined by SAS in the crosstabulation template. After a dynamic variable has been defined, you can use it in the TEXT statement within a footer or header template.

FMISSING
is the number of missing values in the table.

Requirement: The FMISSING dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

NOTITLE
is set to 1 if the PROC FREQ's NOTITLE option was used, and it is set to 0 if the NOTITLE option was not used.

Requirement: The NOTITLE dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

SAMPLESIZE
is set to 0 if the table is empty. Otherwise, it is set to 1.

Requirement: The SAMPLESIZE dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

STRATNUM
is the current stratum number if the table has multiple strata. If the table has only one stratum, then the value is 0.

Requirement: The STRATNUM dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

Featured in: Example 1 on page 460

STRATAVARIABLENAMES
is a string that identifies the current stratum by the name of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined if the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before the dynamic variables can be used in an expression.

STRATAVARIABLELABELS
is a string that identifies the current stratum by the label of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined when the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before you can use the dynamic variables in an expression.

Featured in: Example 1 on page 460 and Example 2 on page 469

NOTES Statement

Provides information about the template.

Tip: The NOTES statement becomes part of the compiled template, which you can view with the SOURCE statement. SAS comments do not become part of the template.

NOTES '*text*';

Required Arguments

'text'
provides information about the template.

END Statement

Ends the cellvalue template.

END;

CELLVALUE Statement

Specifies the order in which the cellvalues are stacked in the cells.

Interaction: If a cellvalue symbol that was specified by the DEFINE CELLVALUE statement is not present in the list, it will not appear in the crosstabulation table.

Featured in: Example 1 on page 460

CELLVALUE *cellvalue(s)*;

Required Arguments

cellvalue(s)
specifies one of the nine possible cellvalues created by the DEFINE CELLVALUE statement. *cellvalues* are ordered from the top to the bottom.

See also: "DEFINE CELLVALUE Statement" on page 441

DEFINE FOOTER Statement

Creates a template for a footer.

Featured in: Example 1 on page 460

DEFINE FOOTER *symbol*;
<*attribute-1*;><*…attribute-n*>;
DYNAMIC *variable-1*<*'text-1'*> <*… variable-n*<*'text-n'*>>;
NOTES *'text'*;
TEXT *header-specification* </ *expression*>;
END;

The substatements in DEFINE FOOTER and the footer attributes are the same as the substatements in DEFINE HEADER and the header attributes. For details about substatements and footer attributes, see the "DEFINE HEADER Statement" on page 449.

DEFINE HEADER Statement

Creates a template for a header.

Featured in: Example 1 on page 460

DEFINE HEADER *symbol*;
<*attribute-1*;><*…attribute-n*>;
DYNAMIC *variable-1*<*'text-1'*> <*… variable-n*<*'text-n'*>>;
NOTES *'text'*;
TEXT *header-specification* </ *expression*>;
END;

Table 9.8 DEFINE HEADER Statements

Task	Statement
Set one or more header attributes	*header-attributes*
Define a symbol that references a value that the data component supplies from the procedure	DYNAMIC
Provide information about the crosstabulation table	NOTES
End a header template	END
Specify the text of the header or the footer	TEXT

Required Arguments

symbol
specifies a name to be referenced by the HEADER statement.

DEFINE HEADER and DEFINE FOOTER Attribute Statements

This section lists all the attributes that you can use in a header or footer template.

Table 9.9 DEFINE HEADER and DEFINE FOOTER Attribute Statements

Task	Attribute
Specify alignment for headers and footers that wrap	CINDENT=
Specify the number of blank lines to place between the current header and the next header or between the current footer and the previous footer	SPACE=
Specify the style element and any changes to its attributes to use for the header or footer	STYLE=

CINDENT='*character*'
specifies alignment for headers or footers that wrap. If a header or footer is too wide to fit on a single line, insert the specified character at the column position at which the second and subsequent lines should start. The first use of the CINDENT character determines the column position. For example, the following TEXT statement makes wrapped lines start at the same column as the left parenthesis:

```
text _COL_NAME_ "(;" _COL_LABEL_ ")"; CINDENT=';';
```

SPACE=*positive-integer*
specifies the number of blank lines to place between the current header and the next header or between the current footer and the previous footer.

Default: 0 for headers and 1 for footers

Tip: The SPACE= attribute is valid only in the LISTING destination.

Featured in: Example 1 on page 460

STYLE=<[style-element-specification(s)]>
specifies the style element and any changes to its attributes to use for the current column. Neither *style-attribute-specification* nor *style-element-name* is required. However, you must use at least one of them.

Note: You can use braces ({ and }) instead of square brackets ([and]). △

style-element-name
is the name of the style element to use to display the data in the column. The style element must be part of a style template that is registered with the Output Delivery System. SAS provides some styles. You can create customized styles with PROC TEMPLATE. For details, see "DEFINE STYLE Statement" on page 490. By default, ODS produces different parts of ODS output with different elements. For example, by default, a table header is displayed with the style element Header. The style elements that you would most likely use with the STYLE= attribute for a table header are as follows:

- Header
- HeaderFixed
- HeaderEmpty
- HeaderEmphasis
- HeaderEmphasisFixed
- HeaderStrong
- HeaderStrongFixed

The style elements that you would most likely use with the STYLE= attribute for a footer are as follows:

- Footer
- FooterFixed
- FooterEmpty
- FooterEmphasis
- FooterEmphasisFixed
- FooterStrong
- FooterStrongFixed

The style element provides the basis for displaying the header or footer. Additional style attributes that you provide can modify the display.

For information on viewing a style template and the available style elements, see "Viewing the Contents of a Style" on page 538. For information about the default style template that ODS uses, see "Working with Styles" on page 538.

style-element-name is either the name of a style element or a variable whose value is a style element.

style-attribute-specification
describes the style attribute to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

For information on the style attributes that you can specify, see "Style Attributes and Their Values" on page 498.

Tip: The STYLE= attribute is valid only in the markup family, printer family, and RTF destinations.

Tip: If you use the STYLE= attribute inside a quoted string, then add a space before or after the carriage return to prevent errors. SAS does not interpret a carriage return as a space. You must explicitly specify spaces in quoted strings.

Featured in: Example 1 on page 460

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Tip: A dynamic variable that is defined in a template is available to that template and all the templates that it contains.

DYNAMIC *dynamic-variable(s)* ;

Required Arguments

dynamic- variable(s)
is a variable that is defined by SAS in the crosstabulation template. After a dynamic variable has been defined, you can use it in the TEXT statement within a footer or header template.

FMISSING
is the number of missing values in the table.

Requirement: The FMISSING dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

NOTITLE
is set to 1 if the PROC FREQ's NOTITLE option was used, and it is set to 0 if the NOTITLE option was not used.

Requirement: The NOTITLE dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

SAMPLESIZE

is set to 0 if the table is empty. Otherwise, it is set to 1.

Requirement: The SAMPLESIZE dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

STRATNUM
is the current stratum number if the table has multiple strata. If the table has only one stratum, then the value is 0.

Requirement: The STRATNUM dynamic variable must be specified by the DYNAMIC statement before you can use the dynamic variable in an expression.

Featured in: Example 1 on page 460

STRATAVARIABLENAMES
is a string that identifies the current stratum by the name of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined if the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before the dynamic variables can be used in an expression.

STRATAVARIABLELABELS
is a string that identifies the current stratum by the label of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined when the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before you can use the dynamic variables in an expression.

Featured in: Example 1 on page 460 and Example 2 on page 469

NOTES Statement

Provides information about the column.

Tip: The NOTES statement becomes part of the compiled column template, which you can view with the SOURCE statement. The SAS comments do not become part of the template.

NOTES '*text*';

Required Arguments

'text'
provides information about the column.

TEXT Statement

Specifies the text of the header or the footer.

Featured in: Example 1 on page 460

TEXT *header-specification(s)* </ *option(s)*>;

Required Arguments

header-specification(s)
specifies the text of the header. *header-specification(s)* can be any dynamic variable that is specified by the DYNAMIC statement, or it can be one of the following:

dynamic-variable
is a variable that is automatically defined by SAS in the crosstabulation template. *dynamic-variable* can be one of the following:

_COL_LABEL_
is the label of the column variable, which is the last variable in a table request.

_COL_NAME_
is the name of the column variable, which is the last variable in a table request. If the column variable does not have a name, then the value of _COL_LABEL_ is an empty text string (“ ”).

_ROW_LABEL_
is the label of the row variable, which is the next to the last variable in a table request. If the row variable does not have a label, the value of _ROW_LABEL_ is an empty text string (“ ”).

_ROW_NAME_
is the name of the row variable, which is the next to the last variable in a table request.

text-specification(s)
specifies the text to use in the header. Each *text-specification* is one of the following:

- a quoted string
- a variable followed by an optional format. The variable is any variable that is declared in a DYNAMIC statement or is any of the variables above.

Tip: If the quoted string is a blank and it is the only item in the header specification, the header is a blank line.

Options

expression
is an expression that is evaluated for a header or footer. If *expression* is omitted, the default is 1. Each DEFINE HEADER statement can contain any number of TEXT statements. The template evaluates each expression in turn from top to bottom and thereby determines the text of the header. The *header-specification* in the first TEXT statement whose expression evaluates to true becomes the header text. After an expression evaluates to true, the template examines no more TEXT statements. If no expression is true, then the header is not used.

expression-1 <comparison-operator expression-n>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value, such as a number or text string.

dynamic variable
is a variable that is defined by SAS in the crosstabulation template or by the DYNAMIC statement within a header or footer template.

_COL_LABEL_
specifies the label of the column variable, which is the last variable in a table request. If the column variable does not have a label, then the value of _COL_LABEL_ is an empty text string (“ ”).

_COL_NAME_
specifies the name of the column variable, which is the last variable in a table request. If the column variable does not have a name, then the value of _COL_NAME_ is an empty text string (“ ”).

FMISSING
is the number of missing values in the table.

Requirement: The FMISSING dynamic variable must be specified by the DYNAMIC statement before you can use it in an expression.

NOTITLE
is set to 1 if PROC FREQ’s NOTITLE option was used, and it is set to 0 if the NOTITLE option was not used.

Requirement: The NOTITLE dynamic variable must be specified by the DYNAMIC statement before you can use it in an expression.

_ROW_LABEL_
is the label of the row variable, which is the next to the last variable in a table request. If the row variable does not have a name, then the value of _ROW_LABEL_ is an empty text string (“ ”).

_ROW_NAME_
specifies the name of the row variable, which is the next to the last variable in a table request. If the row variable does not have a name, then the value of _ROW_NAME_ is an empty text string (“ ”).

SAMPLESIZE
is set to 0 if the table is empty. Otherwise, it is set to 1.

Requirement: The SAMPLESIZE dynamic variable must be specified by the DYNAMIC statement before you can use it in an expression.

STRATNUM
is the current stratum number if the table has multiple strata. If the table has only one stratum, then the value is 0.

Requirement: The STRATNUM dynamic variable must be specified by the DYNAMIC statement before you can use it in an expression.

STRATAVARIABLENAMES
is a string that identifies the current stratum by the name of the stratum variables.

var-1=value-1<var-n=value-n>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined when the table has only one stratum.

Requirement: The STRATAVARIABLENAMES dynamic variable must be specified by the DYNAMIC statement before you can use it in an expression.

STRATAVARIABLELABELS
is a string that identifies the current stratum by the label of the stratum variables. STRATAVARIABLELABELS has the following form:

var-1=value-1<*var-n*=value-n?>

var-1–var-n
specifies the stratum variables.

value-1–value-n
specifies the values of the stratum variables.

Tip: The value is undefined when the table has only one stratum.

Requirement: The DYNAMIC statement must specify the STRATAVARIABLELABELS dynamic variables before you can use them in an expression.

comparison-operator
compares a variable with a value or another variable.

Table 9.10 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

END Statement

END;

END Statement

Ends the crosstabulation template.

Featured in: Example 1 on page 460

END;

Concepts: Crosstabulation Output and the TEMPLATE Procedure

Working with the CrossTabFreqs Crosstabulation Table Template

When creating your own crosstabulation table template, you always define the new table with the same name as the existing table, which is Base.Freq.CrossTabFreqs. By default, the existing crosstabulation table that PROC FREQ creates is stored in the SASHELP.TMPLMST template store.

With PROC TEMPLATE, you can create a modified version of Base.Freq.CrossTabFreqs that you can save in a different template store by using the ODS PATH statement. All crosstabulation templates must have the same name. If you want to have multiple crosstabulation templates, put each one in a different template store. Then you can use the ODS PATH statement to add the template store that contains the version of the crosstabulation template you want to use.

For example, suppose that you have a crosstabulation template in the template store Corporat.Template and another crosstabulation template in Govment.Templat. In the following code, the first ODS PATH statement adds the template store Corporat.Templat. The first PROC FREQ code is then formatted using the crosstabulation table template from Corporat.Templat. The second ODS PATH statement removes Corporat.Templat, and the third ODS PATH statement adds Govment.Templat. The last PROC FREQ step then uses the crosstabulation template from Corporat.Templat.

```
ods path(prepend) corporat.templat(read);
... proc freq code ...
ods path(remove) corporat.templat;
ods path(prepend) govment.templat;
... proc freq code ...
```

For more information about the ODS PATH statement, see “ODS PATH Statement” on page 206.

What Makes the Crosstabulation Table Unique?

Crosstabulation tables produced by PROC FREQ are different from other tables that SAS produces. Most other tables are composed of rows and columns with one value for each row-column combination. However, the crosstabulation table has these distinctive characteristics:

multiple values per cell
: The crosstabulation table can have up to nine values for each row-column combination, depending on the options specified by the TABLES statement. Most other tables that SAS creates have only one value for each row-column combination.

legend
: Crosstabulation tables have a separate box, which is called a legend, to contain the labels for the cellvalues. No other table that SAS produces has a legend.

row variable column
: The far left column contains the row variable values. Each value in this column provides a label for the row in the same way that the column variable values provide labels for the columns.

column variable headers
: Each value in these headers provides a label for the columns of the table.

row and column totals
: The far right column contains the row totals. The bottom row contains the column totals.

grand total cell
: The grand total cell is the last cell of the row and column totals.

Comparison Between Table Templates and Crosstabulation Table Templates

Because the crosstabulation table is unique, the syntax used to create crosstabulation templates differs significantly from other table templates.

- The crosstabulation template has no parent template, and it cannot serve as a parent to another template.
- Most of the attributes, such as CENTER and PANELS=, that are defined for classic table templates are not defined in the crosstabulation table template.
- Crosstabulation table templates use DEFINE CELLVALUE blocks instead of the DEFINE COLUMN blocks that are used in other table templates.
- Crosstabulation table templates use the CELLVALUE statement instead of the COLUMN statement that is used in other table templates.
- Both crosstabulation table templates and other table templates use DEFINE HEADER and DEFINE FOOTER blocks. However, the attributes that can be used in each of these blocks differ.
- DEFINE HEADER and DEFINE FOOTER blocks can contain multiple TEXT statements only in crosstabulation table templates. ODS then chooses which TEXT statement to use at execution time.

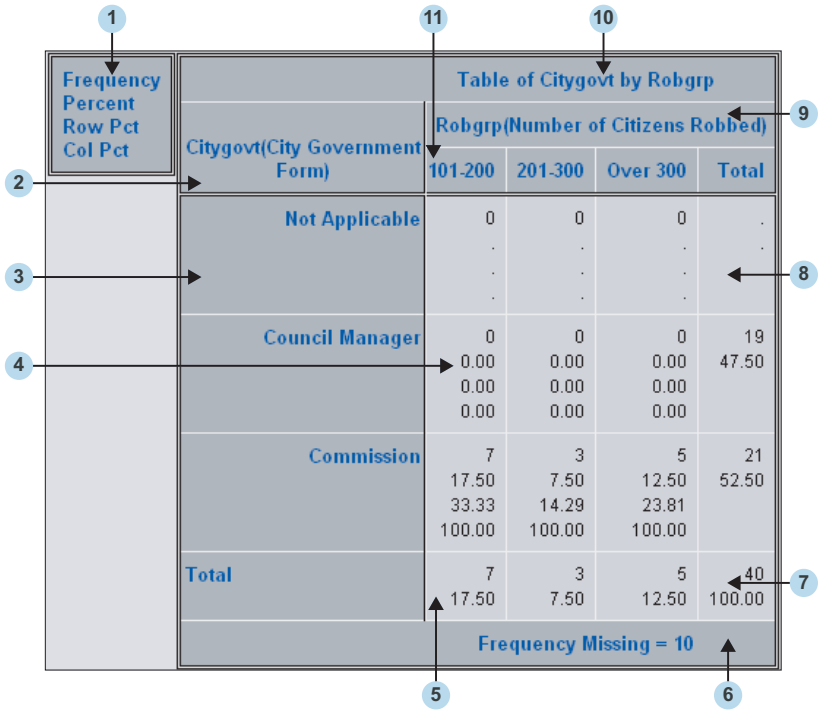
- A TEXT statement can specify a WHERE expression only in crosstabulation table templates. ODS uses the WHERE expression to determine whether to use (the TEXT statement) text as the header text.
- The CELL_STYLE, COLS_HEADER, COL_TOTAL_STYLE, COL_VAR_STYLE, GRAND_TOTAL_STYLE, LEGEND_STYLE, ROWS_HEADER, and ROW_VAR_STYLE attributes are unique to the crosstabulation table.
- The ROWS_HEADER and COLS_HEADER attributes are unique to the crosstabulation table.
- The NVAR, MVAR, and TRANSLATE-INTO statements are not supported for crosstabulation templates.

Crosstabulation Table Regions and Corresponding Attributes

When creating a crosstabulation template, you can use attributes to modify individual table regions. The following figure and corresponding table identify the different parts of the crosstabulation table and the attributes that control the style of each part.

Display 9.3 Crosstabulation Table Regions That Can Be Modified

The FREQ Procedure



Most regions use DEFINE CROSSTABS style attributes to specify a style. The following table show the style attribute that effects each table region. For complete documentation on DEFINE CROSSTABS attributes, see “DEFINE CROSSTABS Attributes” on page 435. Headers and footers use the STYLE= attribute that is valid for the DEFINE HEADER and DEFINE FOOTER statements. For information on the STYLE= attribute, see “DEFINE HEADER and DEFINE FOOTER Attribute Statements” on page 450.

Table 9.11 Table Region and Corresponding Style Attribute

Item	Crosstabulation Table Region	Style Attribute
❶	Legend	LEGEND_STYLE=
❷	Row variable name	ROWS_HEADER=
❸	Row variable value	ROW_VAR_STYLE=
❹	Data cell	CELL_STYLE=
❺	Column total	COL_TOTAL_STYLE=
❻	Footer	STYLE=
❼	Grand total	GRAND_TOTAL_STYLE=
❽	Row total	ROW_TOTAL_STYLE
❾	Column variable name	COLS_HEADER=
❿	Header	STYLE=
⓫	Column variable value	COL_VAR_STYLE=

Examples: Modifying Crosstabulation Output Using the TEMPLATE Procedure

Example 1: Creating a Customized Crosstabulation Table Template with No Legend

PROC TEMPLATE features:

DEFINE CROSSTABS statement:

crosstabs-attributes statements
CELLVALUE statement
DEFINE CELLVALUE statement:

CELLSTYLE AS statement
END statement
FORMAT= attribute
HEADER= attribute
LABEL= attribute

DEFINE HEADER statement:

END statement
SPACE= attribute
STYLE= attribute
TEXT statement

DEFINE FOOTER statement:

END statement
DYNAMIC statement
SPACE= attribute
STYLE= attribute
TEXT statement

END statement
FOOTER statement
HEADER statement
NOTES statement

Other ODS features:

ODS HTML statement
ODS PATH statement
DEFINE STYLE statement

Program Description

The following example creates the crosstabulation table template Base.Freq.CrossTabFreqs. The template has the following features:

- footnote used to display cellvalue labels instead of a legend
- modified headers and footers
- variable labels used in headers
- modified table regions

Program

Create the user-defined formats and create the data set. The FORMAT procedure creates two user-defined formats that can be used in the crosstabulation template. The DATA step creates the Gov data set.

```
Proc Format;
  Value Govtfmt -3='Council Manager'
                 0='Commission'
                 3='Mayor Council'
                .N='Not Applicable'
                 .='   ?';
  Value Robfmt   1='100 or  Less'
                 2='101-200'
                 3='201-300'
                 4='Over 300'
                .N='Not     Known'
                 .='   ?';
  Value Colfg   1='yellow'
                 2='red'
                 3='blue'
                 4='purple'
                .N='green'
                 .='black'
                 other='black';
  Value Rowfg   -3='red'
                 0='purple'
                 3='blue'
                .N='green'
                 .='black'
                 other='black';
```

```
run;

data gov;
   Label Citygovt='City Government Form'
        Robgrp='Number of Meetings Scheduled';
   Input Citygovt Robgrp Weight;  Missing N;
   Format Citygovt Govtfmt. Robgrp Robfmt.;
   LOOP: OUTPUT; WEIGHT=WEIGHT-1; IF WEIGHT>0 THEN GOTO LOOP;
   DROP WEIGHT;
datalines;
0 1  6
0 3  3
0 2  7
0 4  5
N N 10
-3 1 47
-3 3 49
-3 2 63
-3 4 52
. 2  1
3 1 31
3 2 37
3 3 27
3 4 55
3 .  1
;
```

Establish the ODS path and create the White style. The ODS PATH statement specifies the locations to write to or read from when creating the PROC TEMPLATE templates. The PROC TEMPLATE statement, DEFINE STYLE statement, and collection of STYLE statements create the style template White. The ODS NOPROCTITLE statement suppresses the writing of the title of the FREQ procedure.

```
ods path (prepend) work.templat(update);
ods noproctitle;


proc template;
   define style white;
      parent=styles.default;
      style body /
         backgroundcolor=white;
      style systemtitle /
         backgroundcolor=white
         fontsize=6
         fontweight=bold
         fontstyle=italic;
      style systemfooter /
         backgroundcolor=white
         fontsize=2
         fontstyle=italic;
      style proctitle /
```

```
      backgroundcolor=white
      color=#6078bf
      fontweight=bold
      fontstyle=italic;
end;
```

Create the crosstabulation template Base.Freq.CrossTabFreqs. The DEFINE statement creates the crosstabulation template Base.Freq.CrossTabFreqs in the first template store in the path for which you have Write access (Work, in this example). The NOTES statement provides information about the crosstabulation table.

```
define crosstabs Base.Freq.CrossTabFreqs;
   notes "Crosstabulation table";
```

Change the appearance of individual table regions. The following DEFINE CROSSTABS statement attributes modify the appearance of individual table regions. Each attribute corresponds to a specific region of the table. To see which attribute corresponds to which table region, see "Crosstabulation Table Regions and Corresponding Attributes" on page 459.

```
   style=table {backgroundcolor=#BFCFFF};
   cell_style=data {backgroundcolor=#FFFFF0};
   row_var_style=rowheader {backgroundcolor=#BFCFFF  color=rowfg.};
   col_var_style=header {backgroundcolor=#BFCFFF  color=colfg.};
   row_total_style=data {backgroundcolor=#F0F0F0};
   col_total_style=data {backgroundcolor=#F0F0F0};
   grand_total_style=datastrong {backgroundcolor=#F0F0F0};
   legend_style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
```

Specify a row header, a column header, and a label for the table. The ROWS_HEADER= style attribute specifies RowsHeader as the header for rows. The COLS_HEADER= style attribute specifies ColsHeader as the header for columns. The LABEL= attribute specifies a label for the crosstabulation template. The label appears in the Results window.

```
   rows_header=RowsHeader cols_header=ColsHeader;
   label = "Frequency Counts and Percentages";
```

Create the TableOf header template. The DEFINE HEADER statement and its attributes create the header template TableOf, which is specified by the HEADER statement later on in the program.

The TEXT statement specifies the text of the header by using dynamic variables that represent label variables and names. The TEXT statements also use expressions to determine if row labels and column labels are assigned to the row and column variables. Only TEXT statements that have true expressions are displayed in the output. In this example, both the row label and the column label exist. Therefore the first TEXT statement is used and the text resolves to: **"Table of City Government Form by Number of Meetings Scheduled"**.

The STYLE= attribute specifies style information for the header.

```
   define header TableOf;
      text "Table of " _ROW_LABEL_ " by " _COL_LABEL_ / _ROW_LABEL_ ^= ''
```

```
    & _COL_LABEL_ ^= '';
  text "Table of " _ROW_LABEL_ " by " _COL_NAME_  / _ROW_LABEL_ ^= '';
  text "Table of " _ROW_NAME_  " by " _COL_LABEL_ / _COL_LABEL_ ^= '';
  text "Table of " _ROW_NAME_  " by " _COL_NAME_;
  style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
  end;
```

Create the RowsHeader header template. The DEFINE HEADER statement creates the header RowsHeader. RowsHeader is specified as a row header by the preceding ROWS_HEADER= style attribute. The TEXT statements specify the text of the header by using dynamic variables that represent label variables and names. The first TEXT statement uses an expression to determine if a label is assigned to the variable. If there is no label, the next TEXT statement which specifies the row name will be used. In this example there is a row label for the row variable, so in the output, _ROW_LABEL_ resolves to "City Government Form".

The STYLE= attribute specifies style information for the header, and the SPACE attribute specifies that the current header and the previous header should have one blank line between them.

```
define header RowsHeader;
  text _ROW_LABEL_ / _ROW_LABEL_ ^= '';
  text _ROW_NAME_;
  style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
  space=0;
  end;
```

Create the ColsHeader header template. The DEFINE HEADER statement creates the header ColsHeader. ColsHeader is specified as a column header by the preceding COLS_HEADER= style attribute. The TEXT statements specify the text of the header by using dynamic variables that represent label variables and names. The first TEXT statement uses an expression to determine if a label is assigned to the column variable. If there is no label, the next TEXT statement, which specifies the row name, will be used. In this example there is a column label, so in the output, _COL_LABEL_ resolves to "Number of Meetings Scheduled".

The STYLE= attribute specifies style information for the header, and the SPACE attribute specifies that the current header and the previous header should have one blank line between them.

```
define header ColsHeader;
  text _COL_LABEL_ / _COL_LABEL_ ^= '';
  text _COL_NAME_;
  style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
  space=1;
  end;
```

Create the ControllingFor header template. The DEFINE HEADER statement and its attributes create the header template ControllingFor. The DYNAMIC statement declares dynamic variables so that they can be used in expressions. The TEXT statement specifies the text of the header by using dynamic variables that represent label variables and names. In this example, the expression in the TEXT statement resolves to false, so the ControllingFor header does not show up in the output.

The STYLE= attribute specifies style information for the headers.

```
define header ControllingFor;
   dynamic StratNum StrataVariableNames StrataVariableLabels;
   text "Controlling for" StrataVariableNames / StratNum > 0;
   style=header;
   end;
```

Create footer templates. Each of these DEFINE FOOTER statements and its attributes creates a footer template. For the footers to show up in the output, they must be specified by the FOOTER statement.

The DYNAMIC statements declare the dynamic variables FMissing and SampleSize, so that they can be used in the TEXT statements.

The TEXT statements conditionally select text to use as footers. In the first TEXT statement, the expression is true, because FMissing is not 0. Therefore the first TEXT statement is displayed in the output. In the second TEXT statement, the expression resolves to false, so the NoObs footer does not appear in the output.

The STYLE attribute specifies style information for the footers, and the SPACE attribute specifies that the current footer and the previous footer should have one blank line between them.

```
define footer Missing;
   dynamic FMissing;
   text "Frequency Missing = " FMissing -12.99 / FMissing ^= 0;
   style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
   space=1;
   end;

define footer NoObs;
   dynamic SampleSize;
   text "Effective Sample Size = 0" / SampleSize = 0;
   space=1;
   style=header;
   end;
```

Create the cellvalue definitions. The DEFINE CELLVALUE statements define the values that will appear in the cells of the crosstabulation table.

The HEADER= attribute specifies the text that appears in the legend. Because there is no text specified for any of these cellvalues, there is no legend in the output.

The FORMAT= attribute specifies the format to use for the cellvalue. The DATA_FORMAT_OVERRIDE=ON attribute specifies to use the format specified in the data component. The PRINT=ON attribute specifies the cellvalue to appear in the table.

The CELLSTYLE AS statement uses expressions to set the style element of the cells conditionally according to the values of the variables for the Frequency cellvalue. The _VAL_ variable represents the value of a cell. Therefore, in this example, if the value in a cell is less than ten, then the font color for the DataStrong style element is green. If the value in the cell is between 40 and 50, then the font color for the DataStrong style element is orange. If the value is greater then 50, then the font color is red.

```
define cellvalue Frequency;
   header="";
   label="Frequency Count";
   format=BEST7.; data_format_override=on; print=on;
   cellstyle _val_ < 10 as datastrong {color=green},
             _val_ > 40 & _val_ < 50 as datastrong {color=orange},
             _val_ >= 50 as datastrong {color=red};
   end;

define cellvalue Expected;
   header="";
   label="Expected Frequency";
   format=BEST6. data_format_override=on print=on;
   end;

define cellvalue Deviation;
   header="";
   label="Deviation from Expected Frequency";
   format=BEST6. data_format_override=on print=on;
   end;

define cellvalue CellChiSquare;
   header="";
   label="Cell Chi-Square";
   format=BEST6. print=on;
   end;

define cellvalue TotalPercent;
   header="";
   label="Percent of Total Frequency";
   format=6.2 print=on;
   end;

define cellvalue Percent;
   header="";
   label="Percent of Two-Way Table Frequency";
   format=6.2 print=on;
   end;

define cellvalue RowPercent;
   header="";
```

```
      label="Percent of Row Frequency";
      format=6.2 print=on;
      end;

   define cellvalue ColPercent;
      header="";
      label="Percent of Column Frequency";
      format=6.2 print=on;
      end;

   define cellvalue CumColPercent;
      header="";
      label="Cumulative Percent of Column Frequency";
      format=6.2 print=on;
      end;
```

Specify which cellvalues appear in the table and the order in which the cellvalues are stacked in the cells. The CELLVALUE statement specifies which cellvalues appear in the output. In this example, all of the cellvalues that were created appear in the table. The CELLVALUE statement also specifies the order in which the cellvalues are stacked in the cells.

```
   cellvalue
      Frequency Expected Deviation
      CellChiSquare TotalPercent Percent
      RowPercent ColPercent CumColPercent;
```

Specify which headers and footers will appear in the output. The HEADER statement specifies which header templates are applied to your output. The FOOTER statement specifies which footer templates are applied to your output. In order for any of the headers and footers defined by a DEFINE statement to appear in your output, they must be specified by the FOOTER or HEADER statement.

```
   header TableOf ControllingFor;
   footer NoObs Missing;
   end;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. The STYLE= option specifies template White for the output style.

```
ods html file='MyCrosstabsTable.html' style=white;
```

Specify a title and footnote, and suppress the printing of the procedure title. The TITLE and FOOTNOTE statements specify titles and footnotes for the output. The ODS NOPROCTITLE statement prevents the printing of the FREQ procedure's title in the output.

```
title "City Government Form by Number of Meetings Scheduled";
footnote "Cellvalues are stacked in the following order:";
footnote2  "Frequency";
footnote3 "Percent";
```

```
footnote4 "Row Percent";
footnote5 "Column Percent";
ods noproctitle;
```

Create the crosstabulation table. The FREQ procedure creates a Citygovt by Robgrp crosstabulation table.

```
proc freq;
    tables citygovt*robgrp / missprint;
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination, as well as all the files that are open for that destination.

```
ods html close;
```

Output

Display 9.4 Output Using Customized Crosstabulation Table Template

City Government Form by Number of Meetings Scheduled

Table of City Government Form by Number of Meetings Scheduled

City Government Form	Number of Meetings Scheduled						
	?	Not Known	100 or Less	101-200	201-300	Over 300	Total
?	0 . . .	0 . . .	0 . . .	1 . . .	0 . . .	0 . . .	. .
Not Applicable	0 . . .	10 . . .	0 . . .	0 . . .	0 . . .	0 . . .	. .
Council Manager	0 . . .	0 . . .	47 12.30 22.27 55.95	63 16.49 29.86 58.88	49 12.83 23.22 62.03	52 13.61 24.64 46.43	211 55.24
Commission	0 . . .	0 . . .	6 1.57 28.57 7.14	7 1.83 33.33 6.54	3 0.79 14.29 3.80	5 1.31 23.81 4.46	21 5.50
Mayor Council	1 . . .	0 . . .	31 8.12 20.67 36.90	37 9.69 24.67 34.58	27 7.07 18.00 34.18	55 14.40 36.67 49.11	150 39.27
Total	. .	. .	84 21.99	107 28.01	79 20.68	112 29.32	382 100.00

Frequency Missing = 12

Cellvalues are stacked in the following order:
Frequency
Percent
Row Percent
Column Percent

Display 9.5 Output Using Default Crosstabulation Table

City Government Form by Number of Meetings Scheduled

Frequency
Percent
Row Pct
Col Pct

Table of Citygovt by Robgrp

Citygovt(City Government Form)	Robgrp(Number of Meetings Scheduled)						
	?	Not Known	100 or Less	101-200	201-300	Over 300	Total
?	0	0	0	1	0	0	. .
Not Applicable	0	10	0	0	0	0	. .
Council Manager	0 . . .	0 . . .	47 12.30 22.27 55.95	63 16.49 29.86 58.88	49 12.83 23.22 62.03	52 13.61 24.64 46.43	211 55.24
Commission	0 . . .	0 . . .	6 1.57 28.57 7.14	7 1.83 33.33 6.54	3 0.79 14.29 3.80	5 1.31 23.81 4.46	21 5.50
Mayor Council	1 . . .	0 . . .	31 8.12 20.67 36.90	37 9.69 24.67 34.58	27 7.07 18.00 34.18	55 14.40 36.67 49.11	150 39.27
Total	. .	. .	84 21.99	107 28.01	79 20.68	112 29.32	382 100.00

Frequency Missing = 12

Example 2: Creating a Customized Legend

PROC TEMPLATE features:

DEFINE CROSSTABS statement:

crosstabs-attributes statements
CELLVALUE statement
DEFINE CELLVALUE statement:

CELLSTYLE AS statement
END statement
FORMAT= attribute
HEADER= attribute
LABEL= attribute

DEFINE HEADER statement:

END statement
SPACE= attribute
STYLE= attribute
TEXT statement

DEFINE FOOTER statement:

END statement
DYNAMIC statement
SPACE= attribute
STYLE= attribute
TEXT statement

END statement
FOOTER statement
HEADER statement
NOTES statement

Other ODS features:

ODS HTML statement
ODS PATH statement
DEFINE STYLE statement

Program Description

The following example creates a new crosstabulation table template for the CrossTabFreqs table. The template has the following features:

- a legend with customized text
- modified headers and footers
- variable labels used in headers
- modified table regions
- customized styles for cellvalues

Program

Create the user-defined formats and the data set. The FORMAT procedure creates two user-defined formats that can be used in the crosstabulation template. The DATA step creates the Gov data set.

```
Proc Format;
  Value Govtfmt -3='Council Manager'
                 0='Commission'
                 3='Mayor Council'
                .N='Not Applicable'
                 .='   ?';
  Value Robfmt   1='100 or  Less'
                 2='101-200'
                 3='201-300'
                 4='Over 300'
                .N='Not     Known'
                 .='   ?';
  Value colfg   1='yellow'
                 2='red'
                 3='blue'
                 4='purple'
                .N='green'
                 .='black'
                 other='black';
  Value rowfg   -3='red'
                 0='purple'
                 3='blue'
                .N='green'
```

```
                  .='black'
                  other='black';
run;

data gov;
   Label Citygovt='City Government Form'
         Robgrp='Number of Meetings Scheduled';
   Input Citygovt Robgrp Weight;  Missing N;
   Format Citygovt Govtfmt. Robgrp Robfmt.;
   LOOP: OUTPUT; WEIGHT=WEIGHT-1; IF WEIGHT>0 THEN GOTO LOOP;
   DROP WEIGHT;
datalines;
0 1  6
0 3  3
0 2  7
0 4  5
N N 10
-3 1 47
-3 3 49
-3 2 63
-3 4 52
. 2  1
3 1 31
3 2 37
3 3 27
3 4 55
3 .  1
;
```

Establish the ODS path. The ODS PATH statement specifies the locations to write to or read from when you create the PROC TEMPLATE templates.

```
ods path (prepend) work.templat(update);
```

Create the crosstabulation template Base.Freq.CrossTabFreqs. The DEFINE statement creates the crosstabulation template Base.Freq.CrossTabFreqs in the first template store in the path for which you have Write access. The NOTES statement provides information about the crosstabulation table.

```
proc template;
   define crosstabs Base.Freq.CrossTabFreqs;
      notes "Crosstabulation table with legend";
```

Specify a row header, a column header, and a label for the table. The ROWS_HEADER= style attribute specifies RowsHeader as the header for rows. The COLS_HEADER= style attribute specifies ColsHeader as the header for columns. The LABEL= attribute specifies a label for the crosstabulation template. The GRAND_TOTAL_STYLE= changes the FontWeight style attribute in the Data style element to bold. This change affects the values in the rightmost column of the last row in the table.

```
rows_header=RowsHeader cols_header=ColsHeader;
label = "Frequency Counts and Percentages";
grand_total_style=data {fontweight=bold};
```

Create the ControllingFor header template. The DEFINE HEADER statement and its attributes create the header template ControllingFor. The DYNAMIC statement declares dynamic variables so that they can be used in expressions. The TEXT statement specifies the text of the header by using dynamic variables that represent label variables and names. In this example, the expression in the TEXT statement resolves to false, so the ControllingFor header does not show up in the output.

The STYLE= attribute specifies style information for the headers.

```
define header ControllingFor;
   dynamic StratNum StrataVariableNames StrataVariableLabels;
   text "Controlling for" StrataVariableNames / StratNum > 0;
   style=header;
   end;
```

Create the RowsHeader header template. The DEFINE HEADER statement creates the header RowsHeader, which is specified by the preceding ROWS_HEADER= style attribute. The TEXT statements specify the text of the header by using dynamic variables that represent label variables and names. The first TEXT statement uses an expression to determine if a label is assigned to the row variable. If there is no label, the next TEXT statement is used, which specifies the row name. In this example there is a row label for the row variable, so in the output, _ROW_LABEL_ resolves to "City Government Form".

The STYLE= attribute specifies style information for the header. The SPACE= attribute specifies that the current header and the previous header should have one blank line between them. The CINDENT= attribute specifies that wrapped lines start at the same column as the left parenthesis.

```
define header RowsHeader;
   text _ROW_LABEL_ / _ROW_LABEL_ ^= '';
   text _ROW_NAME_;
   space=0;
   style=header;
   cindent=';';
   end;
```

Create the ColsHeader header template. The DEFINE HEADER statement creates the header ColsHeader, which is specified by the preceding COLS_HEADER= style attribute. The TEXT statements specify the text of the header by using dynamic variables that represent label variables and names. The first TEXT statement uses an expression to determine if a label is assigned to the variable. If there is no label, the next TEXT statement is used, which specifies the row name. In this example there is a column label, so in the output, _COL_LABEL_ resolves to "Number of Meetings Scheduled".

The STYLE= attribute specifies style information for the header. The SPACE= attribute specifies that the current header and the previous header should have one blank line between them. The CINDENT= attribute specifies that wrapped lines start at the same column as the left parenthesis.

```
define header ColsHeader;
   text _COL_LABEL_ / _COL_LABEL_ ^= '';
   text _COL_NAME_;
   space=1;
   style=header;
   cindent=';';
   end;
```

Create the TableOf footer template. The DEFINE FOOTER statement and its attributes create the footer template TableOf, which is specified by the FOOTER statement later on in the program.

The TEXT statements specify the text of the header by using dynamic variables that represent label variables and names, the NOTITLE option, and the current stratum number. The TEXT statements use expressions with these variables to determine which text is displayed. Only the TEXT statements that have a true expression are displayed in the output. In this example, the only text statement that has a true expression is the fourth TEXT statement, and the text resolves to: " City Government Form by Number of Meetings Scheduled".

```
define footer TableOf;
   notes 'NoTitle is 1 if the NOTITLE option was specified.';
   dynamic StratNum NoTitle;
   text "Table " StratNum 3. " of " _ROW_LABEL_ " by " _COL_LABEL_ / NoTitle = 0
       & StratNum > 0 & _ROW_LABEL_ ^= '' & _COL_LABEL_ ^= '';
   text "Table " StratNum 3. " of " _ROW_LABEL_ " by " _COL_NAME_  / NoTitle = 0
       & StratNum > 0 & _ROW_LABEL_ ^= ''                    ;
   text "Table " StratNum 3. " of " _ROW_NAME_  " by " _COL_LABEL_ / NoTitle = 0
       & StratNum > 0 & _COL_LABEL_ ^= '';
   text _ROW_LABEL_ " by " _COL_LABEL_ /  NoTitle = 0 &  _ROW_LABEL_ ^= ''
       & _COL_LABEL_ ^= '';
   text _ROW_LABEL_ " by " _COL_NAME_  /  NoTitle = 0 &  _ROW_LABEL_ ^= '';
   text _ROW_NAME_  " by " _COL_LABEL_ /  NoTitle = 0 &  _COL_LABEL_ ^= '';
   text "Table " StratNum 3. " of " _ROW_NAME_  " by " _COL_NAME_  / NoTitle = 0
       & StratNum > 0;
   text _ROW_NAME_  " by " _COL_NAME_ / NoTitle = 0;
   style=header;
   end;
```

Create additional footer templates. Each of these DEFINE FOOTER statements and each of its attributes creates a footer template. To apply these footers to your output, you must specify them in the FOOTER statement.

The DYNAMIC statements declare the dynamic variables FMissing, Stratnum, NoTitle, and SampleSize, so that they can be used in the TEXT statements.

The TEXT statements conditionally select text to use as footers. In the first TEXT statement, the expression is true, because FMissing is not 0. Therefore, the first TEXT statement is displayed in the output. In the second TEXT statement, the expression resolves to false, and the NoObs footer does not appear in the output.

The STYLE attribute specifies style information for the footers. The SPACE attribute specifies that the current footer and the previous footer should have one blank line between them.

```
define footer Missing;
   dynamic FMissing;
   text "Frequency Missing = " FMissing -12.99 / FMissing ^= 0;
   space=1;
   style=header;
   end;

define footer NoObs;
   dynamic SampleSize;
   text "Effective Sample Size = 0" / SampleSize = 0;
   space=1;
   style=header;
   end;
```

Create the cellvalue definitions. The DEFINE CELLVALUE statements define the values that appear in the cells of the crosstabulation table.

The HEADER= attribute specifies the text that appears in the legend. The LABEL= attribute specifies the label for the data set column that corresponds to the cellvalue. The LABEL= attribute affects only the Output destination.

The DATA_FORMAT_OVERRIDE=ON attribute specifies to use the format specified in the data component. The PRINT=ON attribute causes the cellvalue to appear in the table.

The CELLSTYLE AS statement uses expressions to conditionally set the style element of the cells according to the values of the variables for the Percent cellvalue. The _VAL_ variable represents the value of a cell. Therefore, in this example, if the value in a cell is less than ten, then the font color for the DataStrong style element is green. If the value in the cell is greater than twenty, the font color is **#BF6930**.

```
define cellvalue Frequency;
   header="Frequency";
   format=BEST7.;
   label="Frequency Count";
   data_format_override=on print=on;
   end;

define cellvalue Expected;
   header="Expected";
   format=BEST6.;
   label="Expected Frequency";
   data_format_override=on print=on;
   end;
```

```
   define cellvalue Deviation;
      header="Deviation";
      format=BEST6.;
      label="Deviation from Expected Frequency";
      data_format_override=on print=on;
      end;

   define cellvalue CellChiSquare;
      header="Cell Chi-Square";
      format=BEST6.;
      label="Cell Chi-Square";
      print=on;
      end;

   define cellvalue TotalPercent;
      header="Total Percent";
      format=6.2;
      label="Percent of Total Frequency";
      print=on;
      end;

   define cellvalue Percent;
      header="Percent";
      format=6.2;
      label="Percent of Two-Way Table Frequency";
      print=on;
      cellstyle _val_ > 20.0 as {color=#BF6930};
      end;

   define cellvalue RowPercent;
      header="Row Percent";
      format=6.2;
      label="Percent of Row Frequency";
      print=on;
      end;

   define cellvalue ColPercent;
      header="Column Percent";
      format=6.2;
      label="Percent of Column Frequency";
      print=on;
      end;

   define cellvalue CumColPercent;
      header="Cumulative Column Percent";
      format=6.2;
      label="Cumulative Percent of Column Frequency";
      print=on;
end;
```

Specify header and footer templates. The HEADER statement specifies the header templates that are applied to your output. The FOOTER statement specifies the footer templates that are applied to your output. In order for any of the headers and footers that were defined by a DEFINE statement to appear in your output, they must be specified by the FOOTER or HEADER statement.

```
      header ControllingFor;
      footer TableOf NoObs Missing;
```

Specify cellvalues and their order. The CELLVALUE statement specifies which cellvalues will appear in the table and the order. In this example, all of the cellvalues that you created appear in the table, in the order specified by the CELLVALUE statement.

```
   cellvalue
      Frequency Expected Deviation
      CellChiSquare TotalPercent Percent
      RowPercent ColPercent CumColPercent;
end;
run;
```

Specify a title, create the HTML output, and specify the name of the HTML file. The TITLE statement provides a title for the output. The ODS HTML statement opens the HTML destination and creates HTML output. The STYLE= option specifies the style template Ocean for the output.

```
title "City Government Form by Number of Meetings Scheduled";
ods html file='MyCrosstabsTableLegend.html'  style=ocean;
```

Create the crosstabulation table. The FREQ procedure creates a Citygovt by Robgrp crosstabulation table.

```
proc freq;
   tables citygovt*robgrp / missprint;
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are open for that destination.

```
ods html close;
```

Output

Display 9.6 Output Using Customized Crosstabulation Table Template

City Government Form by Number of Meetings Scheduled

The FREQ Procedure

Frequency Percent Row Percent Column Percent	Number of Meetings Scheduled						
City Government Form	?	Not Known	100 or Less	101-200	201-300	Over 300	Total
?	0 . . .	0 . . .	0 . . .	1 . . .	0 . . .	0 . . .	. .
Not Applicable	0 . . .	10 . . .	0 . . .	0 . . .	0 . . .	0 . . .	. .
Council Manager	0 . . .	0 . . .	47 12.30 22.27 55.95	63 16.49 29.86 58.88	49 12.83 23.22 62.03	52 13.61 24.64 46.43	211 55.24
Commission	0 . . .	0 . . .	6 1.57 28.57 7.14	7 1.83 33.33 6.54	3 0.79 14.29 3.80	5 1.31 23.81 4.46	21 5.50
Mayor Council	1 . . .	0 . . .	31 8.12 20.67 36.90	37 9.69 24.67 34.58	27 7.07 18.00 34.18	55 14.40 36.67 49.11	150 39.27
Total	. .	.	84 21.99	107 28.01	79 20.68	112 29.32	382 100.00

City Government Form by Number of Meetings Scheduled

Frequency Missing = 12

Display 9.7 Output Using Default Crosstabulation Table

City Government Form by Number of Meetings Scheduled

Frequency Percent Row Pct Col Pct	Table of Citygovt by Robgrp						
Citygovt(City Government Form)	Robgrp(Number of Meetings Scheduled)						
	?	Not Known	100 or Less	101-200	201-300	Over 300	Total
?	0 . . .	0 . . .	0 . . .	1 . . .	0 . . .	0 . . .	. .
Not Applicable	0 . . .	10 . . .	0 . . .	0 . . .	0 . . .	0 . . .	. .
Council Manager	0 . . .	0 . . .	47 12.30 22.27 55.95	63 16.49 29.86 58.88	49 12.83 23.22 62.03	52 13.61 24.64 46.43	211 55.24
Commission	0 . . .	0 . . .	6 1.57 28.57 7.14	7 1.83 33.33 6.54	3 0.79 14.29 3.80	5 1.31 23.81 4.46	21 5.50
Mayor Council	1 . . .	0 . . .	31 8.12 20.67 36.90	37 9.69 24.67 34.58	27 7.07 18.00 34.18	55 14.40 36.67 49.11	150 39.27
Total	. .	.	84 21.99	107 28.01	79 20.68	112 29.32	382 100.00

Frequency Missing = 12

Example 3: Adding Custom Formats to Cellvalues

PROC TEMPLATE features: EDIT statement

Other ODS features:

ODS HTML statement

ODS PATH statement

Program Description

This example does not use the DEFINE CROSSTABS statement. Instead, it uses the EDIT statement to edit the crosstabulation table template Base.Freq.CrossTabFreqs that was created in Example 2 on page 469 by changing the formats of several cellvalues. In Example 2 on page 469, the following format values were used:

Frequency: BEST6

Percent, RowPercent, ColPercent: 6.2

In this example, the Frequency cellvalue is changed to COMMA12; and the Percent, RowPercent, and ColPercent cellvalues are changed to 6.3.

Program

Create the user-defined formats and the data set. The FORMAT procedure creates four user-defined formats that can be used in the crosstabulation template. The DATA step creates the Gov data set.

```
Proc Format;
  Value Govtfmt -3='Council Manager'
                 0='Commission'
                 3='Mayor Council'
                .N='Not Applicable'
                 .='   ?';
  Value rowfg   -3='red'
                 0='purple'
                 3='blue'
                .N='green'
                 .='black'
                 other='black';
  Value Robfmt   1='100 or  Less'
                 2='101-200'
                 3='201-300'
                 4='Over 300'
                .N='Not     Known'
                 .='   ?';
   Value colfg   1='yellow'
                 2='red'
                 3='blue'
                 4='purple'
                .N='green'
                 .='black'
                 other='black';
```

```
run;

data gov;
   Label Citygovt='City Government Form'
        Robgrp='Number of Meetings Scheduled';
   Input Citygovt Robgrp Weight;  Missing N;
   Format Citygovt Govtfmt. Robgrp Robfmt.;
   LOOP: OUTPUT; WEIGHT=WEIGHT-1; IF WEIGHT>0 THEN GOTO LOOP;
   DROP WEIGHT;
datalines;
0 1  6
0 3  3
0 2  7
0 4  5
N N 10
-3 1 47
-3 3 49
-3 2 63
-3 4 52
. 2  1
3 1 31
3 2 37
3 3 27
3 4 55
3 .  1
;
```

Establish the ODS path. The ODS PATH statement specifies the locations to write to or read from when creating the PROC TEMPLATE templates. The ODS NOPROCTITLE statement suppresses the title of the FREQ procedure.

```
ods noproctitle;
ods path (prepend) work.templat(update);
```

Edit the crosstabulation template Base.Freq.CrossTabFreqs. The EDIT statement changes the crosstabulation table template Base.Freq.CrossTabFreqs that was created in Example 2 on page 469.

```
proc template;
  edit Base.Freq.CrossTabFreqs;
```

Apply new formats to the cellvalues Frequency, Percent, RowPercent, and ColPercent. The FORMAT= attribute specifies a format for the cellvalues. The format COMMA12. is applied to Frequency, and the format 6.3 is applied to Percent, RowPercent, and ColPercent.

```
      edit Frequency;
         format=COMMA12.;

      end;
      edit Percent;
```

```
         format=6.3;
      end;
      edit RowPercent;
         format=6.3;
      end;
      edit ColPercent;
         format=6.3;
      end;
   end;
run;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. The STYLE= option specifies the style template Ocean for the output.

```
ods html file="userfmt.html" style=ocean;
```

Create the crosstabulation table and add a title. The FREQ procedure creates a Citygovt by Robgrp crosstabulation table. The TITLE statement specifies a title.

```
title "Applying Custom Formats to Cellvalues";
proc freq;
   tables citygovt*robgrp / missprint;
run;
```

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination and all the files that are open for that destination.

```
ods html close;
```

Output

Display 9.8 Crosstabulation Output with Custom Formats Applied to Cellvalues

Applying Custom Formats to Cellvalues

Frequency Percent Row Percent Column Percent	Number of Meetings Scheduled						
City Government Form	?	Not Known	100 or Less	101-200	201-300	Over 300	Total
?	0 . . .	0 . . .	0 . . .	1 . . .	0 . . .	0 . . .	. .
Not Applicable	0 . . .	10 . . .	0 . . .	0 . . .	0 . . .	0 . . .	. .
Council Manager	0 . .	0 . .	47 12.304 22.275 55.952	63 16.492 29.858 58.879	49 12.827 23.223 62.025	52 13.613 24.645 46.429	211 55.236
Commission	0 . .	0 . .	6 1.571 28.571 7.143	7 1.832 33.333 6.542	3 0.785 14.286 3.797	5 1.309 23.810 4.464	21 5.497
Mayor Council	1 . .	0 . .	31 8.115 20.667 36.905	37 9.686 24.667 34.579	27 7.068 18.000 34.177	55 14.398 36.667 49.107	150 39.267
Total	. .	.	84 21.990	107 28.010	79 20.681	112 29.319	382 100.00

City Government Form by Number of Meetings Scheduled

Frequency Missing = 12

CHAPTER
10

TEMPLATE Procedure: Creating ODS Graphics

Introduction to the Graph Template Language

Graphics are an indispensable part of statistical analysis. Graphics reveal patterns, identify differences, and provoke meaningful questions about your data. Graphics add clarity to an analytical presentation and stimulate deeper investigation.

SAS 9.2 introduces the Graph Template Language (GTL), a powerful new language for defining clear, effective, statistical graphics. The GTL enables you to generate various types of plots, such as Model Fit plots, Distribution plots, Comparative plots, Prediction Plots, and more.

The GTL applies accepted principles of graphics design to produce plots that are clean and uncluttered. Colors, fonts, and relative sizes of graph elements are all designed for optimal impact. By default, the GTL produces PNG files, which support true color (the full 24-bit RGB color model) and enable visual effects such as anti-aliasing and transparency, but retain a small file size. GTL statement options enable you to control the content and appearance of the plot down to the smallest detail.

The GTL is designed to produce graphics with minimal syntax. The GTL uses a flexible, building-block approach to create a graph by combining statements in a template called a STATGRAPH template. STATGRAPH templates are defined with the TEMPLATE procedure.

You can create custom graphs by defining your own STATGRAPH templates. To create a custom graph, you must

1. define a STATGRAPH template with the TEMPLATE procedure
2. use the Graph Template Language to specify the parameters of your graph
3. associate your data with the template by using the SGRENDER procedure.

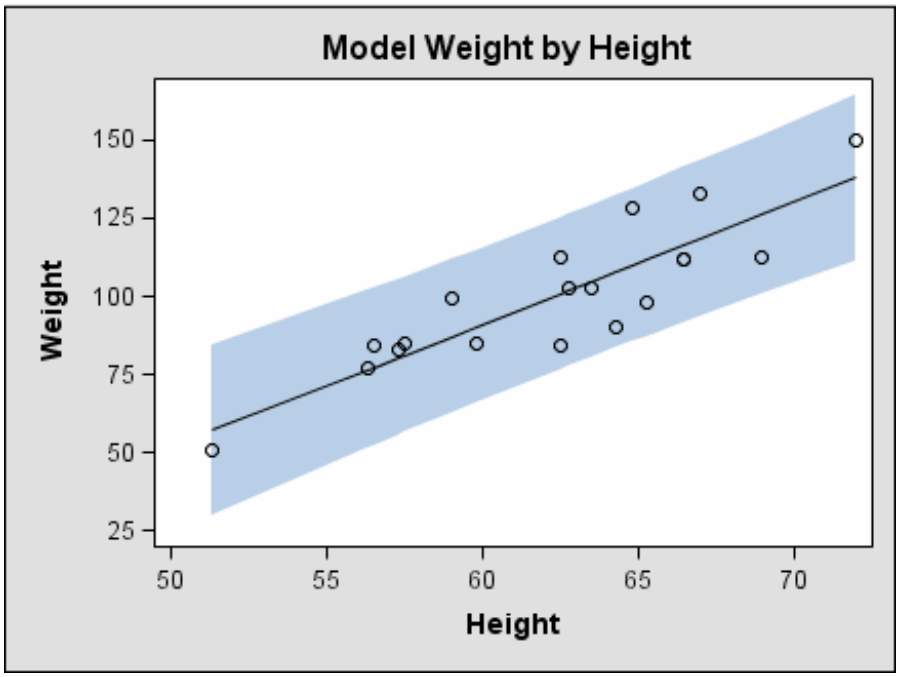
With a few statements, you can create the plots you need to analyze your data. For example, you can create the following Model Fit plot with these statements:

Example Code 10.1 Template For Creating a Model Fit Plot

```
proc template;
define statgraph mytemplate;
beginGraph;
   entrytitle "Model Weight by Height";
```

```
   layout overlay;
   bandplot x=height limitupper=upper limitlower=lower;
   scatterplot y=weight x=height;
   seriesplot y=predict x=height;
endlayout;
endGraph;
end;
run;

proc sgrender data=sashelp.classfit
              template=mytemplate;
run;
```

Display 10.1 Model Fit Plot Using Mytemplate and SASHELP.CLASSFIT



This example defines a STATGRAPH template named **mytemplate**, which uses values from the data set SASHELP.CLASSFIT. This data set contains data variables HEIGHT and WEIGHT and precomputed values for the fitted model (PREDICT) and confidence band (LOWER and UPPER). The SGRENDER procedure uses the data in SASHELP.CLASSFIT and the template **mytemplate** to render the graph. (This example is member GTLMFIT1 in the SAS Sample Library.)

STATGRAPH Syntax: TEMPLATE Procedure

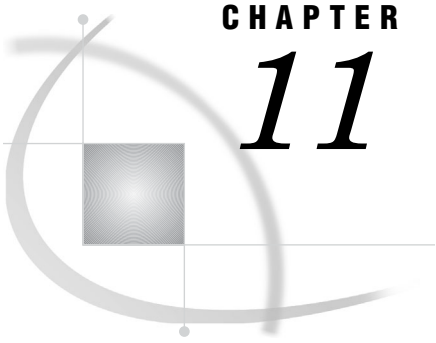
See: For complete documentation on the syntax and usage of the Graph Template Language, see the following documentation:

- *SAS/GRAPH: Graph Template Language Reference*
- *SAS/GRAPH: Graph Template Language User's Guide*

```
PROC TEMPLATE;
   DEFINE STATGRAPH graph-path </ STORE=libref.template-store>;
      DYNAMIC variable-1<'text-1'><variable-n<'text-n'>>;
      MVAR variable-1<'text-1'><variable-n<'text-n'>>;
      NMVAR variable-1<'text-1'><variable-n<'text-n'>>;
      NOTES 'text';
      graph-template-language-statements
      END;
END;
```

Where to Go from Here

- *Creating statistical graphics with ODS:* For reference information about the Graph Template Language, see *SAS/GRAPH: Graph Template Language Reference*.
- *Creating statistical graphics with ODS:* For usage information about PROC TEMPLATE and the Graph Template Language, see *SAS/GRAPH: Graph Template Language User's Guide*.
- *Managing the various templates stored in template stores*: For reference information about the PROC TEMPLATE statements that help you manage and navigate around the many ODS templates, see Chapter 8, "TEMPLATE Procedure: Managing Template Stores," on page 407.
- *Modifying an existing style or creating your own style*: For reference information about the style definition statements in PROC TEMPLATE, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.



CHAPTER

11

TEMPLATE Procedure: Creating a Style Template (Definition)

Overview: ODS Style Templates (Definitions)

Using the TEMPLATE Procedure to Create a Style

The TEMPLATE procedure enables you to customize the look of your SAS output. The TEMPLATE procedure creates and modifies styles. The Output Delivery System then uses these styles to produce customized formatted output.

By default, ODS output is formatted according to the various styles that the procedure or DATA step specify. However, you can also customize the appearance of the output by using the DEFINE STYLE statement in the TEMPLATE procedure.

Terminology

For definitions of terms used in this section, see "Terminology: TEMPLATE Procedure" on page 402.

child
within a dimension hierarchy, a descendant in level n-1 of a member that is at level n. For example, if a Geography dimension includes the levels Country and City, then Bangkok would be a child of Thailand, and Hamburg would be a child of Germany.

parent
within a dimension hierarchy, the ancestor in level n of a member in level n-1. For example, if a Geography dimension includes the levels Country and City, then Thailand would be the parent of Bangkok, and Germany would be the parent of Hamburg. The parent value is usually a consolidation of all of its children's values.

What You Can Do with a Style

Default Style for HTML

By default, ODS uses styles to display the procedure or DATA step results. Modify the appearance of the output by customizing these styles. Display 11.1 on page 488 shows the HTML output from PROC PRINT using the default style. Display 11.2 on page 489 shows the same HTML output from PROC PRINT with a customized style.

Display 11.1 HTML Output from PROC PRINT That Uses the Default Style (Viewed with Microsoft Internet Explorer)

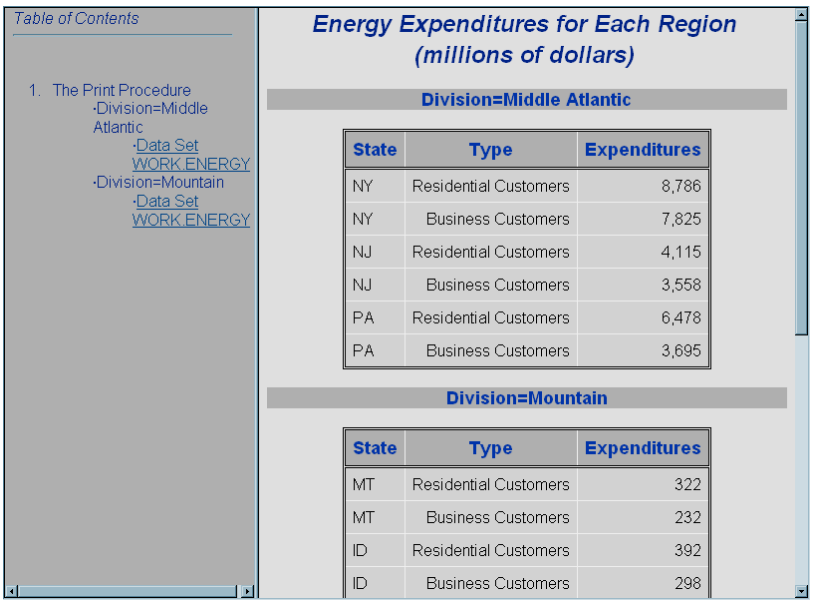


Table of Contents

1. The Print Procedure
   - Division=Middle Atlantic
     - Data Set WORK.ENERGY
   - Division=Mountain
     - Data Set WORK.ENERGY

Energy Expenditures for Each Region (millions of dollars)

Division=Middle Atlantic

State	Type	Expenditures
NY	Residential Customers	8,786
NY	Business Customers	7,825
NJ	Residential Customers	4,115
NJ	Business Customers	3,558
PA	Residential Customers	6,478
PA	Business Customers	3,695

Division=Mountain

State	Type	Expenditures
MT	Residential Customers	322
MT	Business Customers	232
ID	Residential Customers	392
ID	Business Customers	298

Customized Version of the HTML Style

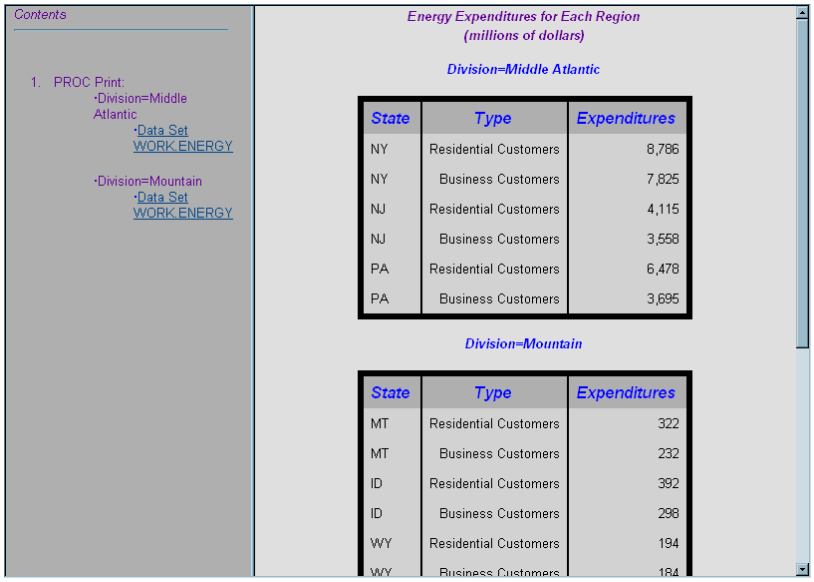
When you are working with styles, you are more likely to modify a SAS style than to write a completely new style. The next display shows the kinds of changes that you can make to the default style for the HTML output. The new style affects both the contents file and the body file in the HTML output. In particular, in the contents file, the style makes changes to the following attributes:

- two of the colors in the color list. One of these colors is the foreground color for the table of contents, the byline, and column headings. The other is the foreground color of many parts of the body file, including SAS titles and footnotes.
- the font size for titles and footnotes.
- the font style for headers.
- the presentation of the data in the table, by changing attributes such as cell spacing, rules, and border width.

In the body file, the new style makes changes to the following attributes:

- the text of the header and the text that identifies the procedure that produced the output
- the colors for some parts of the text
- the font size of some parts of the text
- the spacing in the list of entries in the table of contents

Display 11.2 HTML Output from PROC PRINT with the Customized Style (Viewed with Microsoft Internet Explorer)



Style Syntax: TEMPLATE Procedure

PROC TEMPLATE;

DEFINE STYLE *style-path* | **Base.Template.Style** </ STORE=*libref.template-store*>;
<PARENT=*style-path*;>

NOTES "*text*";
CLASS *style-element-name(s)*<"*text*">
</ *style-attribute-specification(s)*>;
STYLE *style-element-name(s)* <FROM *style-element-name* | _SELF_ > <"*text*">
</ *style-attribute-specification(s)*>;
END;
END;

PROC TEMPLATE Statement

PROC TEMPLATE;
DEFINE STYLE *style-path* | **Base.Template.Style** </ STORE=*libref.template-store*>;
statements-and-attributes
END;

DEFINE STYLE Statement

Creates a style for any destination that supports the STYLE= option.

Requirement: An END statement must be the last statement in the template.
Featured in: Example 1 on page 551

DEFINE STYLE *style-path* | **Base.Template.Style** </ STORE=*libref.template-store*>;
<PARENT=*style-path*;>
NOTES "*text*";
CLASS *style-element-name(s)*<"*text*">
</ *style-attribute-specification(s)*>;
IMPORT *style-specification*<*media-type-1* <, …, *media-type-10*>>
STYLE *style-element-name(s)* <FROM *style-element-name* | _SELF_> <"*text*">
</ *style-attribute-specification(s)*>;
END;

Table 11.1 DEFINE STYLE Statements

Task	Statement
Creates a style element from a style element of the same name	"CLASS Statement" on page 492
Imports Cascading Style Sheet (CSS) information from a file into the style	"IMPORT Statement" on page 492

Task	Statement
Provides information about the style	"NOTES Statement" on page 493
Specifies the style from which the current style inherits its style elements and attributes	"PARENT= Statement" on page 494
Creates or modifies one or more style elements	"REPLACE Statement" on page 494
Ends the style	"REPLACE Statement" on page 494

Required Arguments

style-path
specifies where to store the style. A *style-path* consists of one or more names, separated by periods. Each name represents a directory in a *template store*. PROC TEMPLATE writes the style to the first writable template store in the current path.

Base.Template.Style
creates a style that is the parent of all styles that do not explicitly specify a parent. After this template is created, you do not need to explicitly specify it in your SAS programs. It is automatically applied to all output until you specifically remove it from the item store.

CAUTION:
The Base.Template.Style supplied by SAS contains inheritance information used by many styles. If this inheritance information is not retained, some style elements might not appear in the output. To safely create your own Base.Template.Style, you can start with the existing Base.Template.Style template by writing it to an external file and editing the existing template contents. △

Interaction: The Base.Template.Style master template attributes are overridden by other style templates.

Restriction: If the PARENT= statement is specified, then PARENT= must refer to a style other than Base.Template.Style.

Tip: To view an existing style to base your own Base.Template.Style on, see "Viewing the Contents of a Style" on page 538.

Options

STORE=*libref.template-store*
specifies the template store in which to store the style. If the template store does not exist, then it is created.

Restriction: The syntax of the STORE= option does not become part of the compiled template.

CLASS Statement

Creates a style element from a like-named style element.

Example: The following statements are equivalent:

```
class fonts;
style fonts from fonts;
style fonts from _self_;
```

CLASS *style-element-name(s)* <"*text*"> </ *style-attribute-specification(s)*>;

Required Arguments

style–element-name
specifies one or more style elements to be duplicated and modified.

Tip: If there are multiple style element names specified within a style and an attribute is specified more than once, then the value of the last attribute specified is used.

See: style–element-name in the STYLE statement for a complete description.

See: Appendix 4, "ODS Style Elements," on page 905 for a list of style elements.

Options

The following table lists the options that are available for the CLASS statement. For more detailed descriptions of these options, see the "STYLE Statement" on page 495.

Table 11.2 CLASS Statement Options

Task	Option
Specify new style attributes or modifications to existing style attributes for the new style element	*style-attribute-specification(s)*
Specify information about the CLASS statement	"*text*"

IMPORT Statement

Imports Cascading Style Sheet (CSS) information from a file into the style.

Requirement: CSS files must be written in the same type of CSS that the ODS HTML statement produces. Only class names that match ODS style element names are supported, with no IDs and no context based selectors. To view the CSS code that ODS creates, you can specify the STYLESHEET= option, or you can view the source of an HTML file and look at the code between the tags at the top of the file.

Featured in: Example 6 on page 580

IMPORT *file-specification* <*media-type-1*<, …, *media-type-10*>>

Required Arguments

file-specification
specifies a file, fileref, or URL that contains CSS code. After you import the CSS code, it is converted to style attributes and style elements that can be used with PROC TEMPLATE.

file-specification is one of the following:

"*external-file*"
is the name of the external CSS file.

Requirement: You must enclose *external-file* in quotation marks.

fileref
is a file reference that has been assigned to an external CSS file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see *SAS Language Reference: Dictionary*.

"*URL*"
is a URL to an external CSS file.

Requirement: You must enclose *external-file* in quotation marks.

Options

media-type-1 <, .. media-type-10>
specifies one or more media blocks that correspond to the type of media on which your output will be rendered. CSS uses media type blocks to specify how a document is to be presented on different media—for example, on the screen, on paper, with a speech synthesizer, or with a braille device.

The media block is added to your output in addition to the CSS code that is not contained in any media blocks. By using the *media-type* option, in addition to the general CSS code, you can import the section of a CSS file intended only for a specific media type.

Default: If no *media-type* is specified in your ODS statement, but you have specified media types in your CSS file, then ODS uses the Screen media type.

Range: You can specify up to ten different media types.

Requirement: You must separate multiple *media-types* with commas.

Tip: If you specify multiple media types, all of the style information in all of the media types is applied to your output. However, if there is duplicate style information in different media blocks, then the styles from the last media block are used.

NOTES Statement

Provides information about the style.

Tip: The NOTES statement becomes part of the compiled style template, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES "*text*";

Required Arguments

"text"
provides information about the style.

PARENT= Statement

Specifies the style from which the current style inherits.

PARENT=*style-path*

Required Arguments

style-path
specifies the style to inherit from.

style-path consists of one or more names, separated by periods. Each name represents a directory in a template store. The current style inherits from the specified style in the first readable template store in the current path.

When you specify a parent, all of the style elements, style attributes, and statements that are specified in the parent's style template are used in the current style template unless the current style template overrides them.

SAS provides some styles. You can specify one of these styles for *style-path*, or you can specify a user-defined style. These are some of the styles that are currently shipped with SAS:

- styles.default
- styles.beige
- styles.brick
- styles.brown
- styles.d3d
- styles.minimal
- styles.printer
- styles.statdoc.

For information about finding an up-to-date list of the styles and for viewing a style, see "Viewing the Contents of a Style" on page 538.

Restriction: If the PARENT= statement is specified, then PARENT= must refer to a style other than Base.Template.Style.

REPLACE Statement

The REPLACE statement is no longer supported. Use the "STYLE Statement" on page 495 or "CLASS Statement" on page 492 to create and modify style elements.

STYLE Statement

Creates or modifies one or more style elements.

Featured in: Example 1 on page 551

STYLE *style-element-name(s)*
<FROM *existing-style-element-name* | _SELF_><"*text*">
</ *style-attribute-specification(s)*>;

Required Arguments

style-element-name
specifies one or more style elements to be created or modified. If *style-element-name* is a new style element, then PROC TEMPLATE stores the style element in the current style. If *style-element-name* overrides a style element that is a parent of another element, then all of the descendents of *style-element-name*, including those inherited from parent styles, also inherit the new attributes.

Tip: If a like-named style element already exists in the child style and it *is not* created by using the FROM option, then the style element in the child style overrides the style element of the same name in the parent style.

Tip: If a like-named style element already exists in the child style and it *is* created by using the FROM option, then the style attributes from the parent style element are absorbed into the style element in the child style.

Tip: If an attribute is defined in a like-named style element in the parent style and it is not explicitly specified in the STYLE statement of the new like-named style element, then the attribute is not inherited, unless you specify the FROM option.

Tip: If there are multiple identical style element names specified within a style and an attribute is specified more then once, then the value of the last attribute specified is used.

Requirement: Style elements must be separated by commas.

Example: The following STYLE statement ❶, which uses a style element list, is equivalent to STYLE statements ❷-❹:

```
❶style data, data1, dataempty from _self_ /
        color = red
        backgroundcolor = black;


❷style data  from data /
        color = red
        backgroundcolor = black;
❸style data1 from data1/
        color = red
        backgroundcolor = black;
```

```
❹style dataempty from dataempty /
        color = red
        backgroundcolor = black
```

See also: style-element-name on page 495

See: Appendix 4, "ODS Style Elements," on page 905 for a list of style elements.

Featured in: Example 5 on page 576

Options

FROM *existing-style-element-name* | _SELF_
specifies that the preceding *style-element-name* inherit the style attributes from the *existing-style-element-name*.

existing-style-element-name
specifies the existing style element that another style element inherits from. *existing-style-element-name* can have the same name as the preceding *style-element-name*, or it can be the name of another style element. The style element must exist in the current style or in the parent of the current style.

Tip: If a like-named style element already exists in the child style and it *is not* created by using the FROM option, then the style element in the child style overrides the style element of the same name in the parent style.

Tip: If a like-named style element already exists in the child style and it *is* created by using the FROM option, then the style attributes from the parent style element are absorbed into the style element in the child style.

Tip: If an attribute is defined in a like-named style element in the parent style and it is not explicitly specified in the STYLE statement of the new like-named style element, then the attribute is not inherited, unless you specify the FROM option.

Tip: PROC TEMPLATE looks first in the current style for the style element. If PROC TEMPLATE does not find the style element, then it looks in the parent style.

Example: The following statement specifies that the style element Data2 be created from the style element Data1, and that the COLOR=BLACK style attribute be added.

```
style data1 from data2 / color=black;
```

SELF
specifies that the parent of the style element should have the same name as the new style element.

Tip: The _SELF_ option is most useful when specifying multiple style elements.

Example: The following STYLE statement ❶ is equivalent to STYLE statements ❷-❹:

```
❶style data, data1, dataempty from _self_ /
        color = red
        backgroundcolor = black;


❷style data from  data /
        color = red
        backgroundcolor = black;
❸style data1 from  data1 /
```

```
            color = red
            backgroundcolor = black;
        ❹style dataempty from dataempty /
            color = red
            backgroundcolor = black
```

See: Appendix 4, "ODS Style Elements," on page 905 for a list of style elements.

style-attribute-specification(s)
specifies new style attributes or modifications to existing style attributes for the new style element. Each *style-attribute-specification* has this general form:

style-attribute-name=< | >*style-attribute-value*

style-attribute-name
is the name of an attribute that is listed in "Style Attributes and Their Values" on page 498, or it is the name of a user-defined style attribute.

Restriction: If *style-attribute-name* refers to a user-defined attribute, then enclose the name in quotation marks. If *style-attribute-name* refers to an attribute that is listed in "Style Attributes and Their Values" on page 498, then do not enclose the name in quotation marks.

style-attribute-value
assigns the value to the attribute. If an attribute from the list in "Style Attributes and Their Values" on page 498 is specified, then specify the kind of value that the attribute expects.

For more information about style-attribute values, see "Style Attributes and Their Values" on page 498.

|
prevents the style attribute from being inherited by any child style elements.

Restriction: If there are multiple style element names specified within a style and an attribute is specified more than once, then the value of the last attribute specified is used.

Tip: Override any attribute of the parent style element, whether it is inherited or explicitly defined, by specifying it in the STYLE statement without the FROM option.

Tip: If an attribute is defined in a like-named style element in the parent style and it is not explicitly specified in the STYLE statement of the new like-named style element, then the attribute is not inherited, unless you specify the FROM option.

"*text*"
provides information about the STYLE statement. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not become part of the compiled style.

END Statement

Ends the style.

END;

Style Attributes and Their Values

Style attributes influence the characteristics of individual cells, tables, documents, graphs, and HTML frames.

See also: For information about using style attributes with ODS Statistical Graphics, see the chapter on controlling the appearance of your graphics in *SAS/GRAPH: Graph Template Language User's Guide*.

See also: For a table of style elements that can be used with style attributes, see Appendix 4, "ODS Style Elements," on page 905.

See also: For more information about using style attributes and style elements together, see "Understanding Styles, Style Elements, and Style Attributes" on page 540.

See also: For information about style attribute values, see "Style Attribute Values" on page 534.

Style Attributes Overview

Style attributes exist within style elements and are specified by the "STYLE Statement" on page 495 or the "CLASS Statement" on page 492. The default value for an attribute depends on the style that is in use. For information about styles, style elements, and style attributes, see "Understanding Styles, Style Elements, and Style Attributes" on page 540.

Style attributes can be supplied by SAS or user-defined. Style attributes can be referenced with a style reference. See the *style-reference* value in the section "Style Attribute Values" on page 534 for more information.

The implementation of an attribute depends on the ODS destination that formats the output. When creating HTML output, the implementation of an attribute depends on the browser that is used. For information about viewing the attributes in a style, see "Viewing the Contents of a Style" on page 538.

For a list of the values that style attributes can specify, see "Style Attribute Values" on page 534. For a list of style elements that you can specify style attributes in, see Appendix 4, "ODS Style Elements," on page 905.

Style AttributesTables

Table 11.3 Table of General Style Attributes

Attribute	Task	Destinations	Affected Items
ABSTRACT= on page 509	Specify whether styles used in an HTML document are used in CSS or LaTex style files	Markup family	HTML documents
ACTIVELINKCOLOR= on page 509	Specify the color that a link in an HTML document changes to after you click it, but before the browser opens that file	Markup family	HTML documents
ASIS= on page 509	Specify how to handle leading spaces and line breaks in an HTML document	Markup family, printer family, and RTF	Cells and HTML documents
BACKGROUNDCOLOR= on page 509	Specify the color of the background of tables, cells, or graphs	Markup family, printer family, and RTF	Cells, tables, graphs
BACKGROUNDIMAGE= on page 510	Specify an image to use as the background	Markup family, PCL, and PS	Cells, tables, graphs
BACKGROUNDREPEAT= on page 510	Specify whether an image is repeated horizontally, vertically, both, or not repeated	Markup family	Individual tables or cells, graphs
BODYSCROLLBAR= on page 510	Specify whether to put a scroll bar in the frame that references the body file	Markup family	Individual frames in HTML output
BODYSIZE= on page 510	Specify the width of the frame that displays the body file in the HTML frame file	Markup family	Individual frames in HTML output
BORDERBOTTOMCOLOR= on page 511	Specify the color of the bottom border of the table	Markup family, printer family, RTF, and Measured RTF	Bottom border of a table
BORDERBOTTOMSTYLE= on page 511	Specify the line style of the bottom border of the selected cell	Markup family, RTF, and Measured RTF	Bottom border of a cell

Attribute	Task	Destinations	Affected Items
BORDERBOTTOMWIDTH= on page 511	Specify the width of the bottom border of the table	Markup family, printer family, RTF, and Measured RTF	Bottom border of a table
BORDERCOLOR= on page 511	Specify the color of the border in a table or cell if the border is just one color	Markup family, printer family, RTF, and Measured RTF	Individual tables or cells
BORDERCOLORDARK= on page 511	Specify the darker color to use in a border that uses two colors to create a three-dimensional effect	Markup family and printer family	Individual tables or cells
BORDERCOLORLIGHT= on page 512	Specify the lighter color to use in a border that uses two colors to create a three-dimensional effect	Markup family and printer family	Individual tables or cells
BORDERLEFTCOLOR= on page 512	Specify the color of the left border of a table	Markup family, printer family, RTF, and Measured RTF	Left border of the table
BORDERLEFTSTYLE= on page 512	Specify the line style of the left border of the specified cell	Markup family, RTF, and Measured RTF	Left border of the specified cell
BORDERLEFTWIDTH= on page 512	Specify the width of the left border of the table	Markup family, printer family, RTF, and Measured RTF	Left border of a table
BORDERRIGHTCOLOR= on page 512	Specify the color of the right border of the table	Markup family, printer family, RTF, and Measured RTF	Right border of a table
BORDERRIGHTSTYLE= on page 513	Specify the line style of the right border of the selected cell	Markup family, RTF, and Measured RTF	Right border of the selected cell
BORDERRIGHTWIDTH= on page 513	Specify the width of the right border of the table	Markup family, printer family, RTF, and Measured RTF	Right border of the table

Attribute	Task	Destinations	Affected Items
BORDERTOPCOLOR= on page 513	Specify the color of the top border of the table	Markup family, printer family, RTF, and Measured RTF	Top border of the table
BORDERTOPSTYLE= on page 513	Specify the line style of the top border of the specified cell	Markup family, RTF, and Measured RTF	Top border of the specified cell
BORDERTOPWIDTH= on page 514	Specify the width of the top border of the table	Markup family, printer family, RTF, and Measured RTF	Top border of the table
BORDERWIDTH= on page 514	Specify the width of the border of the table	Markup family, RTF, printer family	Individual tables or cells
CELLPADDING= on page 514	Specify the amount of white space on each of the four sides of the text in a cell in the table	Markup family, RTF, printer family	Tables
CELLSPACING= on page 514	Specify the thickness of the spacing between cells in a table	Markup family, RTF, printer family	Tables
CLASS= on page 514	Specify the name of the style sheet class to use in an HTML document for the table or cell	Markup family	Individual tables or cells
COLOR= on page 515	Specify the color of the foreground in tables, cells, or graphs, which is primarily the color of text	Markup family, RTF, printer family	Individual tables or cells, and graphs
CONTENTPOSITION= on page 515	Specify the position, within the frame file, of the frames that display the contents and the page files	Markup family	Individual frames in HTML output
CONTENTSCROLLBAR= on page 515	Specify whether to put a scroll bar in the frames in the frame file that display the contents and the page files	Markup family	Individual frames in HTML output

Attribute	Task	Destinations	Affected Items
CONTENTSIZE= on page 516	Specify the width of the frames in the frame file that display the contents and the page files	Markup family	Individual frames in HTML output
CONTENTTYPE= on page 516	Specify the value of the content type for pages in an HTML document that is sent directly to a web server rather than to a file	Markup family	Individual frames in HTML output
CONTRASTCOLOR= on page 516	Specify the alternate colors for maps	Markup family, RTF, printer family	Graphs
DOCTYPE= on page 517	Specify the entire doctype declaration for the HTML document	Markup family	HTML documents
FILLRULEWIDTH= on page 517	Place a rule of the specified width into the space around the text (or entire cell if there is no text) in a table where white space would otherwise appear	printer family	HTML documents
FLYOVER= on page 518	Specify the text to show in a data tip for the cell	Markup family, PDF	Individual cells
FONT= on page 518	Specify a font definition to use in tables, cells, and graphs	Markup family, RTF, printer family	Individual tables or cells, graphs
FONTFAMILY= on page 518	Specify the font to use in cells and graphs	Markup family, RTF, printer family	Individual tables or cells graphs
FONTSIZE= on page 518	Specify the size of the font for tables, cells, and graphs	Markup family, RTF, printer family	Individual tables or cells graphs
FONTSTYLE= on page 519	Specify the style of the font for tables, cells, and graphs	Markup family, RTF, printer family	Individual tables or cells graphs

Attribute	Task	Destinations	Affected Items
FONTWEIGHT= on page 519	Specify the font weight of tables, cells, and graphs	Markup family, RTF, printer family	Individual tables or cells graphs
FONTWIDTH= on page 519	Specify the font width of tables, cells, and graphs compared to the width of the usual design of the table, cell, or graph	Markup family, RTF, printer family	Individual tables or cells graphs
FRAME= on page 520	Specify the type of frame to use on a table	Markup family, RTF, printer family	Tables
FRAMEBORDER= on page 520	Specify whether to put a border around the frame for an HTML file that uses frames	Markup family	Individual frames in HTML output
FRAMEBORDERWIDTH= on page 520	Specify the width of the border around the frames for an HTML file that uses frames	Markup family	Individual frames in HTML output
FRAMESPACING= on page 521	Specify the width of the space between frames for HTML that uses frames	Markup family	Individual frames in HTML output
HEIGHT= on page 521	Specify the height of a cell, graph, or graphics in an HTML document [1]	Markup family, RTF, printer family	Cells, HTML documents, and graphs
HREFTARGET= on page 521	Specify the window or frame in which to open the target of the link	Markup family	Individual cells
HTMLID= on page 522	Specify an id for the table or cell	Markup family	Individual tables or cells
HTMLSTYLE= on page 522	Specify individual attributes and values for a table or cell in an HTML document	Markup family	Individual tables or cells
IMAGE= on page 522	Specify the image to appear in a graph	Markup family, printer family, and RTF	Graphs
LINKCOLOR= on page 523	Specify the color for the links in an HTML document that have not yet been visited	Markup family, printer family, and RTF	HTML documents

Attribute	Task	Destinations	Affected Items
LISTENTRYANCHOR= on page 523	Specify whether to make the entry in the table of contents a link to the body file	Markup family	HTML documents
LISTENTRYDBLSPACE= on page 524	Specify whether to double space between entries in the table of contents	Markup family	HTML documents
LISTSTYLETYPE= on page 524	Specify the string to use for the bullets in the contents file	Markup family	Individual frames in HTML output
MARGINBOTTOM= on page 525	Specify the bottom margin for the HTML document	Markup family, printer family, and RTF	HTML documents
MARGINLEFT= on page 525	Specify the left margin for the HTML document	Markup family, printer family, and RTF	HTML documents
MARGINRIGHT= on page 526	Specify the right margin for the HTML document	Markup family, printer family, and RTF	HTML documents
MARGINTOP= on page 526	Specify the top margin for the HTML document	Markup family, printer family, and RTF	HTML documents
NOBREAKSPACE= on page 526	Specify how to handle space characters	Markup family, printer family, and RTF	Individual cells
OVERHANGFACTOR= on page 526	Specify an upper limit for extending the width of the column in an HTML document	Markup family and printer family	HTML documents
PAGEBREAKHTML= on page 526	Specify HTML to place at page breaks in an HTML document	Markup family	Tables, cells, and HTML documents
POSTHTML on page 527	Specify the HTML code to place after the table or cell	Markup family	Individual tables or cells
POSTIMAGE= on page 527	Specify an image to place before the table or cell	Markup family	Individual tables or cells
POSTTEXT= on page 527	Specify text to place after the cell or table	Markup family, printer family, and RTF	Individual tables or cells

Attribute	Task	Destinations	Affected Items
PREHTML= on page 527	Specify the HTML code to place before the table or cell	Markup family	Individual tables or cells
PREIMAGE= on page 527	Specify an image to place before the table or cell	Markup family, printer family, and RTF	Individual tables or cells
PRETEXT= on page 528	Specify text to place before the cell or table	Markup family, printer family, and RTF	Individual tables or cells
PROTECTSPECIALCHARACTERS= on page 528	Specify how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted in cells	Markup family, printer family, and RTF	Individual tables or cells
RULES= on page 528	Specify the types of rules to use in tables	Markup family, printer family, and RTF	Tables
STARTCOLOR= on page 529	Specify the start fill color for a graph	HTML	Graphs
TAGATTR= on page 529	Specify text to insert in the HTML	Markup family	Individual cells
TEXTALIGN= on page 529	Specify justification in tables, cells, and graphs	printer family and RTF	Individual tables or cells graphs
TEXTDECORATION= on page 531	Change the visual presentation of the text	Markup family, RTF, and printer family	Individual tables or cells
TEXTINDENT= on page 531	Specify the number of spaces that the first line of output will be indented	Markup family, RTF, and printer family	Individual tables or cells
TEXTJUSTIFY= on page 532	Specify if the words of the text are to be spaced evenly or if the characters are to be evenly justified	HTML, RTF, and TAGSETS.RTF	Titles, footnotes, and text
TRANSPARENCY= on page 532	Specify a transparency level for graphs	HTML	Graphs
URL= on page 532	Specify a URL to link to	Markup family, RTF, and printer family	Individual cells
VERTICALALIGN= on page 532	Specify vertical justification	Markup family, printer family, and RTF	Individual cells and graphs

Attribute	Task	Destinations	Affected Items
VISITEDLINKCOLOR= on page 533	Specify the color for links that have been visited in an HTML document	Markup family	HTML documents
WATERMARK= on page 533	Specify whether to make the image that is specified by BACKGROUNDIMAGE= into a "watermark "	Markup family	HTML documents
WIDTH= on page 533	Specify the width of a cell, table, line, or a graph	Markup family, printer family, and RTF	Tables

[1] This attribute can also be used to influence other characteristics as described in another section of the table

Note: You can use the value _UNDEF_ for any style attribute. ODS treats an attribute that is set to _UNDEF_ as if its value had never been set, even in the parent or beyond. △

Graphical style attributes can be used in graphical style elements for device-based graphics or template-based graphics (ODS graphics). Different style attributes are valid for different style elements. For a table of style elements and the style attributes that are valid in each one, see “Style Elements Affecting Template-Based Graphics” on page 914 and “Style Elements Affecting Device-Based Graphics” on page 920.

Device-based graphics are all SAS/GRAPH output where there is a user-specified or default device (DEVICE= option) that controls certain aspects of the graphical output. Supplied device drivers are stored in the Sashelp.Devices catalog. Examples of devices drivers are SASPRTC, GIF, WIN, ACTIVEX, PDF, and SVG. Common SAS/GRAPH procedures that produce device-based graphics are GPLOT, GCHART, and GMAP. Most device-based graphics produce a GRSEG catalog entry as output and use the GOPTIONS statement to control the graphical environment.

Template-based graphics graphics include all SAS/GRAPH output where a compiled ODS template of type STATGRAPH is used to produce graphical ouput. Supplied templates are stored in Sashelp.Tmplmst. Device drivers and some global statements such as SYMBOL, PATTERN, AXIS, and LEGEND have no affect on this form of graphics. Common SAS/GRAPH procedures that produce template-based graphics are SGPLOT, SGPANEL, and SGRENDER, in addition to many SAS/STAT, SAS/ETS, and SAS/QC procedures. ODS graphics always produce ouput as image files and use the ODS GRAPHICS statement to control the graphical environment.

Table 11.4 Table of Graphical Style Attributes

Attribute	Task	Graphics Environment	Affected Items
BACKGROUNDIMAGE= on page 510	Specify an image file path	Device-based graphics	Image that can be stretched, but not positioned in graph, chart, walls, floor
CAPSTYLE= on page 514	Specify the shape of the line at the end of a box whisker graph	Template-based graphics	Shape of line at end of box whisker
COLOR= on page 515	Specify the color of the foreground in tables, cells, or graphs, which is primarily the color of text	All graphics environments	Background color of the graph, walls, or floor; color of text
CONNECT= on page 515	Specify characteristics of a box plot connect line	Template-based graphics	Box plot connect line
CONTRASTCOLOR= on page 516	Specify the color a line or marker	Template-based graphics	Color of line or marker
DISPLAYOPTS= on page 516	Specify display features for graphs	Template-based graphics	Displayed features of box plots, ellipses, histograms, bands
DROPSHADOW= on page 517	Specify whether the drop shadow color for text is displayed	Device-based graphics	Drop shadow color for text
ENDCOLOR= on page 517	Specify the final color used with a two or three color ramp	All graphics environments	Contours, gradient legends
FONT= on page 518	Specify a font definition to use in tables, cells, and graphs	All graphics environments	All text font attributes
FONTFAMILY= on page 518	Specify the font to use in cells and graphs	All graphics environments	Font family
FONTSIZE= on page 518	Specify the size of the font for tables, cells, and graphs	All graphics environments	Font size
FONTSTYLE= on page 519	Specify the style of the font for tables, cells, and graphs	All graphics environments	Font style

Attribute	Task	Graphics Environment	Affected Items
FONTWEIGHT= on page 519	Specify the font weight of tables, cells, and graphs	All graphics environments	Font weight
FRAMEBORDER= on page 520	Specify whether there is a graph wall border	All graphics environments	Graph wall border
GRADIENT_DIRECTION= on page 521	Specify the direction of the gradient	Device-based graphics	Graph background, legend background, charts, walls, floors
IMAGE= on page 522	Specify the path to an image	Device-based graphics	Image that can be positioned, but not stretched in graph, chart, walls, floor
LINESTYLE= on page 523	Specify the pattern of a line	All graphics environments	Borders, axis lines, grid, reference
LINETHICKNESS= on page 523	Specify the thickness of a line	All graphics environments	Thickness of line
MARKERSIZE= on page 525	Specify a marker size	All graphics environments	Marker size
MARKERSYMBOL= on page 525	Specify a marker symbol	All graphics environments	Marker used
NEUTRALCOLOR= on page 526	Specify the middle color of a 3–color ramp	Template-based graphics	Contours, gradient legends
OUTPUTHEIGHT= on page 526	Specify the height of a graph	All graphics environments	Height of graph
OUTPUTWIDTH= on page 526	Specify the width of a graph	All graphics environments	Width of graph
STARTCOLOR= on page 529	Specify the start fill color for a graph	All graphics environments	Contours, gradient legends
TEXTALIGN= on page 529	Specify the alignment of an image	Device-based graphics	Image horizontal positioning
TICKDISPLAY= on page 532	Specify the placement of all major and minor axis tick marks	Template-based graphics	Placement of all axis tick marks, major and minor

Attribute	Task	Graphics Environment	Affected Items
TRANSPARENCY= on page 532	Specify the transparency of backgrounds, fills, lines, and markers	All graphics environments	Backgrounds, fills, lines, markers
VERTICALALIGN= on page 532	Specify vertical justification	Device-based graphics	Image vertical positioning

Detailed Information for All Style Attributes

ABSTRACT= ON | OFF
specifies whether styles used in an HTML document are used in CSS or LateX style files.

ON
specifies that styles are used in CSS or LateX style files.

OFF
specifies that styles are not used in CSS or LateX style files.

Restriction: The ABSTRACT= attribute is valid only in markup family destinations.

ACTIVELINKCOLOR=*color*
specifies the color that a link in an HTML document changes to after you click it, but before the browser opens that file.

Restriction: The ACTIVELINKCOLOR= attribute is valid only in markup family destinations.

See: color on page 534

ASIS=ON|OFF
specifies how to handle leading spaces and line breaks in an HTML document.

ON
prints text with leading spaces and line breaks, in the same manner as the listing output.

OFF
trims leading spaces and ignores line breaks.

Default: OFF

Restriction: The ASIS= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

BACKGROUNDCOLOR= *color*
specifies the color of the background of the tables, cells, or graphs.

Alias: BACKGROUND=

Interaction: The CBACK= option in the SAS/GRAPH GOPTIONS statement overrides the BACKGROUNDCOLOR= attribute.

Restriction: The BACKGROUNDCOLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: Generally, the background color of the cell overrides the background color of the table. You see the background color for the table only as the space between cells (see CELLSPACING= on page 514).

See: color on page 534

Featured in: Example 1 on page 551 and Example 3 on page 564

BACKGROUNDIMAGE="*string*"
specifies an image in a table, cell, or graph to use as the background. Viewers can tile or stretch the image as the background for the HTML table or graph that the procedure creates. For graphs, the specified image is stretched.

string
is the name of a GIF or JPEG file. Use a simple file name, a complete path, or a URL. However, the most versatile approach is to use a simple filename and to place all image files in the local directory.

Restriction: The BACKGROUNDIMAGE= attribute is valid only in markup family destination, the PCL destination, and the PS destination.

Interaction: The BACKGROUNDIMAGE= attribute is overridden by the IBACK= and IMAGESTYLE=FIT options in the SAS/GRAPH GOPTIONS statement.

See: string on page 537

BACKGROUNDREPEAT= REPEAT | REPEAT_X | REPEAT_Y | NO_REPEAT
specifies whether an image is repeated horizontally, vertically, both, or not repeated.

NO_REPEAT
specifies that the image is not repeated.

REPEAT
specifies that the image is repeated both horizontally and vertically.

REPEAT_X
specifies that the image is repeated horizontally.

REPEAT_Y
specifies that the image is repeated vertically.

Restriction: The BACKGROUNDREPEAT= attribute is valid only in markup family destinations.

BODYSCROLLBAR=YES | NO | AUTO
specifies whether to put a scroll bar in the frame that references the body file.

YES
places a scroll bar in the frame that references the body file.

NO
specifies not to put a scroll bar in the frame that references the body file.

AUTO
places a scroll bar in the frame that references the body file only if needed.

Tip: Typically, BODYSCROLLBAR= is set to AUTO.

Restriction: The BODYSCROLLBAR= attribute is valid only in markup family destinations.

BODYSIZE= *dimension* | *dimension%* | *
specifies the width of the frame that displays the body file in the HTML frame file.

dimension
is a nonnegative number or the width of the frame specified as a percentage of the entire display.

*

specifies to use whatever space is left after displaying the content and page files as specified by the CONTENTSIZE= attribute.

Tip: If *dimension* is a nonnegative number then the unit of measure is pixels.

Restriction: The BODYSIZE= attribute is valid only in markup family destinations.

See: dimension on page 535

See also: For information about the HTML files that ODS creates, see "HTML Links and References Produced by the HTML Destination" on page 891.

BORDERBOTTOMCOLOR=*color*

specifies the color of the bottom border of the table.

Restriction: The BORDERBOTTOMCOLOR= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

See also: color on page 534

BORDERBOTTOMSTYLE= *line-style*

specifies the line style of the bottom border of the specified cell.

line-style

can be one of the following:

DASHED

DOTTED

DOUBLE

GROOVE

HIDDEN

INSET

OUTSET

RIDGE

SOLID

Restriction: The BORDERBOTTOMSTYLE= attribute is valid only in markup family destinations, RTF destination, and the Measured RTF destination.

BORDERBOTTOMWIDTH=*dimension*

specifies the width of the bottom border of the table.

Restriction: The BORDERBOTTOMWIDTH= attribute is valid only in markup family destinations, RTF destination, printer family destinations, and the Measured RTF destination.

See: dimension on page 535

BORDERCOLOR= *color*

specifies the color of the border in a table or cell if the border is just one color.

Restriction: The BORDERCOLOR= attribute is valid only in markup family destinations, RTF destination, printer family destinations, and the Measured RTF destination.

See also: color on page 534

BORDERCOLORDARK= *color*

in a table or cell , specifies the darker color to use in a border that uses two colors to create a three-dimensional effect.

Interaction: The BORDERCOLORDARK style attribute is ignored in HTML4 output because it is not part of the HTML4 standard. To create a color border in the HTML4 output, use the BORDERCOLOR= style attribute.

Restriction: The BORDERCOLORDARK= attribute is valid only in markup family destinations and printer family destinations.

See: color on page 534

Featured in: Example 4 on page 570

BORDERCOLORLIGHT= *color*
in a table or cell, specifies the lighter color to use in a border that uses two colors to create a three-dimensional effect.

Interaction: The BORDERCOLORLIGHT style attribute is ignored in the creation of HTML4 output because it is not part of the HTML4 standard. To create a color border in HTML4 output, use the BORDERCOLOR= style attribute.

Restriction: The BORDERCOLORLIGHT= attribute is valid only in markup family destinations and printer family destinations.

See: color on page 534

Featured in: Example 4 on page 570

BORDERLEFTCOLOR=*color*
specifies the color of the left border of the table.

Restriction: The BORDERLEFTCOLOR= attribute is valid only in markup family destinations, RTF destination, printer family destinations, and the Measured RTF destination.

See also: color on page 534

BORDERLEFTSTYLE= *line-style*
specifies the line style of the left border of the specified cell.

line-style
can be one of the following:

DASHED
DOTTED
DOUBLE
GROOVE
HIDDEN
INSET
OUTSET
RIDGE
SOLID

Restriction: The BORDERLEFTSTYLE= attribute is valid only in markup family destinations, RTF destination, and the Measured RTF destination.

BORDERLEFTWIDTH=*dimension*
specifies the width of the left border of the table.

Restriction: The BORDERLEFTWIDTH= attribute is valid only in markup family destinations, RTF destination, printer family destinations, and the Measured RTF destination.

See: dimension on page 535

BORDERRIGHTCOLOR=*color*
specifies the color of the right border of the table.

Restriction: The BORDERRIGHTCOLOR= attribute is valid only in markup family destinations, RTF destination, printer family destinations, and the Measured RTF destination.

See also: color on page 534

BORDERRIGHTSTYLE= *line-style*
specifies the line style of the right border of the selected cell.

line-style
can be one of the following:

DASHED
DOTTED
DOUBLE
GROOVE
HIDDEN
INSET
OUTSET
RIDGE
SOLID

Restriction: The BORDERRIGHTSTYLE= attribute is valid only in markup family destinations, RTF destination, and the Measured RTF destination.

BORDERRIGHTWIDTH=*dimension*
specifies the width of the right border of the table.

Restriction: The BORDERRIGHTWIDTH= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

See: dimension on page 535

BORDERTOPCOLOR=*color*
specifies the color of the top border of the table.

Restriction: The BORDERTOPCOLOR= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

Restriction: To ensure that the top border color is created, specify the BORDERTOPWIDTH= and the BORDERTOPCOLOR= attribute for the RTF destination.For the RTF destination, specify the BORDERTOPCOLOR= attribute in conjunction with the BORDERTOPWIDTH= attribute to ensure that the top border color is created.

See also: color on page 534

BORDERTOPSTYLE= *line-style*
specifies the line style of the top border of the specified cell.

line-style
can be one of the following:

DASHED
DOTTED
DOUBLE
GROOVE
HIDDEN
INSET

OUTSET
RIDGE
SOLID

Restriction: The BORDERTOPSTYLE= attribute is valid only in markup family destinations, the RTF destination, and the Measured RTF destination.

Restriction: For the RTF destination, specify the BORDERTOPSTYLE= attribute in conjunction with the BORDERTOPWIDTH= attribute to ensure that the style of the top border is the style that you specified.

BORDERTOPWIDTH=*dimension*
specifies the width of the top border of the table.

Restriction: The BORDERTOPWIDTH= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

See: dimension on page 535

BORDERWIDTH= *dimension*
specifies the width of the border of the table.

Restriction: The BORDERWIDTH= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: Typically, when BORDERWIDTH=0, the ODS destination sets RULES=NONE (see the discussion about RULES= on page 528) and FRAME=VOID (see the discussion about FRAME= on page 520).

Featured in: Example 1 on page 551 and Example 3 on page 564

See: dimension on page 535

CAPSTYLE= "SERIF" | "LINE" | "BRACKET" | "NONE"
specifies the shape of the line at the end of a box whisker.

CELLPADDING=*dimension* | *dimension*%
specifies the amount of white space on each of the four sides of the text in a cell in the table.

dimension
is a nonnegative number or the amount of white space on each of the four sides of the text in a cell specified as a percentage of the table.

Restriction: The CELLPADDING= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: dimension on page 535

Featured in: Example 3 on page 564

CELLSPACING=*dimension*
specifies the thickness of the spacing between cells in a table.

Interaction: If BORDERWIDTH= is nonzero, and if the background color of the cells contrasts with the background color of the table, then the color of the cell spacing is determined by the table's background.

Restriction: The CELLSPACING= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: dimension on page 535

Featured in: Example 1 on page 551 and Example 3 on page 564

CLASS="*string*"
specifies the name of the style sheet class to use in an HTML document for the table or cell.

Alias: HTMLCLASS=

Restriction: The CLASS= attribute is valid only in markup family destinations.

See: string on page 537

COLOR=*color*
specifies the color of the foreground in tables, cells, or graphs, which is primarily the color of text.

Alias: FOREGROUND=

Interaction: The COLOR= attribute is overridden by the CBACK= option in the SAS/GRAPH GOPTIONS statement.

Restriction: The COLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: In a table, the COLOR= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

See: color on page 534

Featured in: Example 3 on page 564

CONNECT= "MEDIAN" | "MEAN" | "Q1" | "Q3" | "MIN" | "MAX"
specifies the characteristics of a box plot connect line.

CONTENTPOSITION= LEFT | RIGHT | TOP | BOTTOM
specifies the position, within the frame file, of the frames that display the contents and the page files.

LEFT
places the frames on the left.

Alias: L

RIGHT
places the frames on the right.

Alias: R

TOP
places the frames at the top.

Alias: T

BOTTOM
places the frames at the bottom.

Alias: B

Restriction: The CONTENTPOSITION= attribute is valid only in markup family destinations.

See also: For information about the HTML files that ODS creates, see "HTML Links and References Produced by the HTML Destination" on page 891.

CONTENTSCROLLBAR=YES | NO |AUTO
specifies whether to put a scroll bar in the frames in the frame file that display the contents and the page files. (For information about the HTML files that ODS creates, see "HTML Links and References Produced by the HTML Destination" on page 891.)

YES
places a scroll bar in the frames in the frame file that display the contents and the page files.

NO
specifies not to put a scroll bar in the frames in the frame file that display the contents and the page files.

AUTO
specifies

Tip: Typically, CONTENTSCROLLBAR= is set to AUTO.

Restriction: The CONTENTSCROLLBAR= attribute is valid only in markup family destinations.

See also: For information about the HTML files that ODS creates, see "HTML Links and References Produced by the HTML Destination" on page 891.

CONTENTSIZE=*dimension* | *dimension* % | *
specifies the width of the frames in the frame file that display the contents and the page files.

dimension
is a nonnegative number or the width of the frames specified as a percentage of the entire display.

*
specifies to use whatever space is left after displaying the body file as specified by the BODYSIZE= attribute.

Requirement: *dimension* % must be a positive number between 0 and 100.

Tip: If *dimension* is a nonnegative number, then the unit of measure is pixels.

Restriction: The CONTENTSIZE= attribute is valid only in markup family destinations.

See: dimension on page 535

See also: BODYSIZE= on page 510

See also: For information about the HTML files that ODS creates, see "HTML Links and References Produced by the HTML Destination" on page 891

CONTENTTYPE="*string*"
specifies the value of the content type for pages in an HTML document that is sent directly to a web server rather than to a file.

string
is the content type for the pages.

Requirement: *string* must be enclosed in quotation marks.

Tip: The value of *string* is usually "text/html".

See: string on page 537

Alias: HTMLCONTENTTYPE=

Restriction: The CONTENTTYPE= attribute is valid only in markup family destinations.

CONTRASTCOLOR=*color*
specifies the alternate colors for maps. The alternate colors are applied to the blocks on region areas in block maps.

Restriction: The CONTRASTCOLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: color on page 534

DISPLAYOPTS= "CAPS" | "CONNECT" | "FILL" | "MEAN" | "MEDIAN" | "NOTCHES" | "OUTLIERS" | "OUTLINE"
specifies one or more display features for ODS graphs. To specify multiple features, enclose the list of features in quotation marks, for example:

```
displayopts="fill caps mean''
```

CAPS
displays caps at the ends of the whiskers.

Restriction: CAPS can be used only for box plots.

CONNECT
displays the line connecting multiple boxes.

Restriction: CONNECT can be used only for box plots.

FILL
displays filled boxes, bars, ellipses, and bands.

Restriction: FILL can be used only for box plots, histograms, ellipses, and confidence bands.

MEAN
displays the mean symbol within a box.

Restriction: MEAN can be used only for box plots.

MEDIAN
displays the median line within the box.

NOTCHES
displays notched boxes.

Restriction: NOTCHES can be used only for box plots.

OUTLIERS
displays markers for the outliers.

Restriction: OUTLIERS can be used only for box plots.

OUTLINE
displays outlined ellipses and bars.

Restriction: OUTLINE can be used only for ellipses, bands, and histograms.

DOCTYPE="*string*"
specifies the entire doctype declaration for the HTML document, including the opening "<!DOCTYPE" and the closing ">".

string
is the doctype declaration.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

Alias: HTMLDOCTYPE=

Restriction: The DOCTYPE= attribute is valid only in markup family destinations.

DROPSHADOW= ON | OFF
specifies whether the drop shadow color for text is displayed.

ENDCOLOR=*color*
specifies the final color used with a two or three color ramp.

See: color on page 534

FILLRULEWIDTH= *dimension*
places a rule of the specified width into the space around the text (or entire cell if there is no text) in a table where white space would otherwise appear.

Tip: If no text is specified, then FILLRULEWIDTH= fills the space around the text with dash marks. For example: –this– or this ——.

Restriction: The FILLRULEWIDTH= attribute is valid only in printer family destinations.

See: dimension on page 535

FLYOVER="*string*"
specifies the text to show in a data tip for the cell.

string
is the text of the data tip.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

Restriction: The FLYOVER= attribute is valid only in markup family destinations and the PDF destination.

FONT=*font-definition*
specifies a font definition to use in tables, cells, and graphs.

Tip: For a table, the FONT= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

Tip: If the system does not recognize the font specified, then it will refer to the system's default font. This attribute does not accept concatenated fonts. SAS Graph Styles can only specify one font.

Restriction: The FONT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Featured in: Example 3 on page 564

See: font-definition on page 536

FONTFAMILY="*string-1*<*..., string-n*>"
specifies the font to use in cells and graphs. If you supply multiple fonts, then the destination device uses the first one that is installed on the system.

string
is the name of the font.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

Alias: FONT_FACE=

Restriction: The FONTFAMILY= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: For a table, the FONTFAMILY= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

Tip: You cannot be sure what fonts are available to someone who is viewing the output in a browser or printing it on a high-resolution printer. Most devices support the following fonts:

- Times
- Courier
- Arial, Helvetica.

Featured in: Example 1 on page 551

FONTSIZE=*dimension* | *size*
specifies the size of the font for tables, cells, and graphs.

dimension
is a nonnegative number.

Alias: FONT_SIZE=

Restriction: If you specify a dimension, then specify a unit of measure. Without a unit of measure, the number becomes a relative size.

See: dimension on page 535

size
The value of *size* is relative to all other font sizes in the HTML document.

Range: 1 to 7

Tip: For a table, the FONTSIZE= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

Restriction: The FONTSIZE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Featured in: Example 1 on page 551

FONTSTYLE= ITALIC | ROMAN | SLANT
specifies the style of the font for tables, cells, and graphs. In many cases, italic and slant map to the same font.

Alias: FONT_STYLE=

Restriction: The FONTSTYLE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: For a table, the FONTSTYLE= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

Featured in: Example 1 on page 551 and Example 3 on page 564

FONTWEIGHT= *weight*
specifies the font weight of tables, cells, and graphs. *weight* is any of the following:

- MEDIUM
- BOLD
- DEMI_BOLD
- EXTRA_BOLD
- LIGHT
- DEMI_LIGHT
- EXTRA_LIGHT.

Alias: FONT_WEIGHT=

Restriction: You cannot be sure what font weights are available to someone who is viewing the output in a browser or printing it on a high-resolution printer. Most devices support only MEDIUM and BOLD, and possibly LIGHT.

Restriction: The FONTWEIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: For a table, the FONTWEIGHT= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

Featured in: Example 1 on page 551

FONTWIDTH=*relative-width*
specifies the font width of tables, cells, and graphs compared to the width of the usual design of the table, cell, or graph. *relative-width* is any of the following:

- NORMAL

- COMPRESSED
- EXTRA_COMPRESSED
- NARROW
- WIDE
- EXPANDED.

Alias: FONT_WIDTH=

Restriction: Few fonts honor these values.

Restriction: The FONTWIDTH= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: For a table, the FONTWIDTH= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a cell.

Featured in: Example 1 on page 551

FRAME=*frame-type*
specifies the type of frame to use on a table. This table shows the possible values for *frame-type* and their meanings:

Table 11.5 Frame-type Values

Value for *frame-type*	Frame Type
ABOVE	A border at the top
BELOW	A border at the bottom
BOX	Borders at the top, bottom, and both sides
HSIDES	Borders at the top and bottom
LHS	A border at the left side
RHS	A border at the right side
VOID	No borders
VSIDES	Borders at the left and right sides

Restriction: The FRAME= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Featured in: Example 3 on page 564

FRAMEBORDER=ON | OFF
specifies whether to put a border around the frame for an HTML file that uses frames.

ON
places a border around the frame for an HTML file that uses frames.

OFF
specifies not to put a border around the frame for an HTML file that uses frames.

Restriction: The FRAMEBORDER= attribute is valid only in markup family destinations.

FRAMEBORDERWIDTH=*dimension*
specifies the width of the border around the frames for an HTML file that uses frames.

Restriction: The FRAMEBORDERWIDTH= attribute is valid only in markup family destinations.

See: dimension on page 535

FRAMESPACING=*dimension*
specifies the width of the space between frames for HTML that uses frames.

Restriction: The FRAMESPACING= attribute is valid only in markup family destinations.

See: dimension on page 535

GRADIENT_DIRECTION= "YAXIS" | "XAXIS "
specifies the direction of the gradient.

"YAXIS"
specifies a vertical gradient.

"XAXIS"
specifies a horizontal gradient.

HEIGHT=*dimension*
specifies the height of a cell, graph, or graphics in an HTML document.

dimension
is a nonnegative number.

See: dimension on page 535

Alias: CELLHEIGHT=

Alias: OUTPUTHEIGHT=

Restriction: The HEIGHT= option does not apply to output generated as a result of GRSEG (graph segment) output.

Interaction: The YPIXELS= option in the SAS/GRAPH GOPTIONS statement overrides the HEIGHT= attribute.

Tip: HTML automatically sets cell height appropriately. You will seldom need to specify this attribute in the HTML destination.

Restriction: The HEIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

HREFTARGET=*target*
specifies the window or frame in which to open the target of the link. *target* is one of these values:

_BLANK
opens the target in a new, blank window. The window has no name.

_PARENT
opens the target in the window from which the current window was opened.

_SEARCH
opens the target in the browser's search pane.

Restriction: Only available in Internet Explorer 5.0 or later.

_SELF
opens the target in the current window.

_TOP
opens the target in the topmost window.

"*name*"
opens the target in the specified window or the frame.

Default: _SELF

Restriction: The HREFTARGET= attribute is valid only in markup family destinations.

HTMLID="*string*"
specifies an ID for the table or cell. The ID is for use by a Java Script.

string
is the ID text.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

Restriction: The HTMLID= attribute is valid only in markup family destinations.

HTMLSTYLE="*string*"
specifies individual attributes and values for a table or cell in an HTML document.

string
is the name of an attribute or value.

Requirement: *string* must be enclosed in quotation marks.

See: *string on page 537*

Restriction: The HTMLSTYLE= attribute is valid only in markup family destinations.

IMAGE="*string*"
specifies the image to appear in a graph. This image is positioned or tiled.

string
is the name of the image.

Requirement: *string* must be enclosed in quotation marks.

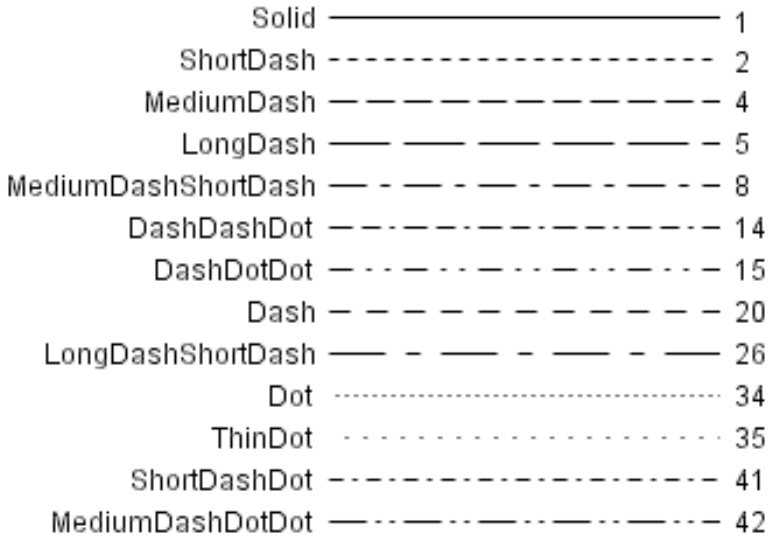
See: string on page 537

Interaction: The BACK= and IMAGESTYLE=TILE options in the SAS/GRAPH GOPTIONS statement override the IMAGE= attribute.

Restriction: The IMAGE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

LINESTYLE=*pattern-number*
specifies the pattern of a line.Valid pattern numbers range from 1 to 46. Not all pattern numbers have names.You must specify the line pattern by it's number. *pattern-number* can be one of the following:

Display 11.3 Table of Line Patterns



LINETHICKNESS=*dimension*
specifies the thickness of a line.

See: dimension on page 535

LINKCOLOR=*color*
specifies the color for the links in an HTML document that have not yet been visited.

Restriction: The LINKCOLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: color on page 534

LISTENTRYANCHOR=ON | OFF
in an HTML document, the LISTENTRYANCHOR= attribute specifies whether to make the entry in the table of contents a link to the body file.

ON
specifies to make this entry in the table of contents a link to the body file.

OFF
specifies not to make this entry in the table of contents a link to the body file.

Restriction: The LISTENTRYANCHOR= attribute is valid only in markup family destinations.

LISTENTRYDBLSPACE=ON | OFF
in an HTML document, the LISTENTRYDBLSPACE= attribute specifies whether to double space between entries in the table of contents.

ON
specifies to double space between entries in the table of contents.

OFF
specifies not to double space between entries in the table of contents.

Restriction: The LISTENTRYDBLSPACE= attribute is valid only in markup family destinations.

LISTSTYLETYPE=*string*
specifies the string to use for the bullets in the contents file. ODS uses bullets in the contents file.

string
is one of the following:

- circle
- decimal
- disc
- lower_alpha
- lower_roman
- none
- square
- upper_alpha
- upper_roman.

Alias: BULLET

See: string on page 537

Restriction: The LISTSTYLETYPE= attribute is valid only in markup family destinations.

MARKERSIZE=*dimension*
specifies the marker size (both width and height).

See: dimension on page 535

MARKERSYMBOL=*marker-symbol*
specifies a marker symbol. *marker-symbol* can be one of the following:

Display 11.4 Table of Marker Symbols

↓	ArrowDown	▽	HomeDown	∿	Tilde	●	CircleFilled
✳	Asterisk	⌶	Ibeam	△	Triangle	◆	DiamondFilled
○	Circle	+	Plus	∪	Union	▼	HomeDownFilled
◇	Diamond	□	Square	×	X	■	SquareFilled
>	GreaterThan	☆	Star	Y	Y	★	StarFilled
#	Hash	⊤	Tack	Z	Z	▲	TriangleFilled

MARGINBOTTOM= *dimension*
specifies the bottom margin for the HTML document.

Alias: BOTTOMMARGIN=

Restriction: The MARGINBOTTOM= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: dimension on page 535

MARGINLEFT=*dimension*
specifies the left margin for the HTML document.

Alias: LEFTMARGIN=

Restriction: The MARGINLEFT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: dimension on page 535

MARGINRIGHT=*dimension*
specifies the right margin for the HTML document.

Alias: RIGHTMARGIN=

Restriction: The MARGINRIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: dimension on page 535

MARGINTOP= *dimension*
specifies the top margin for the HTML document.

Alias: TOPMARGIN=

Restriction: The MARGINTOP= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See: dimension on page 535

NEUTRALCOLOR=*color*
specifies the middle color in a 3-color ramp.

See: color on page 534

NOBREAKSPACE= ON | OFF
specifies how to handle space characters in cells.

ON
does not let SAS break a line at a space character.

OFF
lets SAS break a line at a space character if appropriate.

Restriction: The NOBREAKSPACE= attribute is valid in markup family destinations, printer family destinations, and the RTF destination.

OUTPUTHEIGHT=*dimension*
specifies the height of a graph.

See: dimension on page 535

OUTPUTWIDTH=*dimension*
specifies the width of a graph.

See: dimension on page 535

OVERHANGFACTOR= *nonnegative-number*
specifies an upper limit for extending the width of the column in an HTML document.

Tip: Typically, an overhang factor between 1 and 2 works well.

Tip: The HTML that is generated by ODS tries to ensure that the text in a column wraps when it reaches the requested column width. When the overhang factor greater than 1, the text can extend beyond the specified width.

Restriction: The OVERHANGFACTOR= attribute is valid only in markup family and printer family destinations.

PAGEBREAKHTML= "*string*"
specifies HTML to place at page breaks in an HTML document.

string
is the HTML code used to place at page breaks.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

Restriction: The PAGEBREAKHTML= attribute is valid only in markup family destinations.

POSTHTML= "*string*"
specifies the HTML code to place after the table or cell.

string
is the HTML code to place after a table or cell.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

Restriction: The POSTHTML= attribute is valid only in markup family destinations.

Featured in: Example 3 on page 564

POSTIMAGE= "*string*" | *fileref*
specifies an image to place before the table or cell.

string
names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

Requirement: *string* must be enclosed in quotation marks.

See: string on page 537

fileref
is a reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref. (

See: "Statements" in *SAS Language Reference: Dictionary* for information about the FILENAME statement.

Restriction: The POSTIMAGE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

POSTTEXT= "*string*"
specifies text to place after the cell or table.

Requirement: *string* must be enclosed in quotation marks.

Restriction: The POSTTEXT= attribute is valid only for markup family destinations, printer family destinations, and the RTF destination.

See: string on page 537

PREHTML="*string*"
specifies the HTML code to place before the table or cell.

Restriction: The PREHTML= attribute is valid only for markup family destinations.

See: string on page 537

PREIMAGE= "*string*" | *fileref*
specifies an image to place before the table or cell.

string
names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

Requirement: Enclose *string* in quotation marks.

See: string on page 537

fileref
is a reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref. (For information about the FILENAME statement, see "Statements" in *SAS Language Reference: Dictionary*.)

Restriction: The PREIMAGE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

PRETEXT="*string*"
specifies text to place before the cell or table.

string
text that is placed before the cell or table.

Requirement: Enclose *string* in quotation marks.

See: string on page 537

Restriction: The PRETEXT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

PROTECTSPECIALCHARACTERS=ON | OFF | AUTO
specifies how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted in cells. In HTML and other markup languages, these characters indicate the beginning of a markup tag, the end of a markup tag, and the beginning of the name of a file or character entity.

ON
interprets special characters as the characters themselves. That is, when ON is in effect the characters are protected before they are passed to the HTML or other markup language destination so that the characters are not interpreted as part of the markup language. Using ON enables you to show markup language tags in the HTML document.

OFF
interprets special characters as markup language tags. That is, when OFF is in effect, the characters are passed to the HTML or other markup language destination without any protection so that the special characters are interpreted as part of the markup language.

AUTO
interprets any string that starts with a < and ends with a > as a markup language tag (ignoring spaces that immediately precede the <, spaces that immediately follow the >, and spaces at the beginning and end of the string). In any other string, AUTO protects the special characters from their markup language meaning.

Restriction: The PROTECTSPECIALCHARACTERS= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

RULES=*rule-type*
specifies the types of rules to use in tables. This table shows the possible values for the RULES= attribute and their meanings:

Table 11.6 RULES= Attribute Values

Value of RULES= Attribute	Locations of Rules
ALL	Between all rows and columns
COLS	Between all columns
GROUPS	Between the table header and the table and between the table and the table footer, if there is one
NONE	No rules anywhere
ROWS	Between all rows

Restriction: The RULES= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Featured in: Example 4 on page 570

STARTCOLOR= ***color***
specifies the start fill color for a graph. It is used to create a gradient effect.

Note: You can have either a start and end gradient effect or no gradient effect. If you specify a TRANSPARENCY level and you only specify the STARTCOLOR, then the end color will be completely transparent gradationally to the specified start color. △

Restriction: The STARTCOLOR= attribute is valid only for the HTML destination.

See: color on page 534

TAGATTR="*string*"
specifies text to insert into HTML.

string
is the text that is inserted into HTML tags.

Requirement: *string* must be enclosed in quotation marks.

Requirement: *string* must be valid HTML for the context in which the style element is created.

Tip: Many style elements are created between <TD> and </TD> tags. To determine how a style element is created, look at the source for the output.

See: string on page 537

Restriction: The TAGATTR= attribute is valid only in markup family destinations.

TEXTALIGN= CENTER | DEC | LEFT | RIGHT
specifies justification in tables, cells, and graphs. In graphs, this option specifies the justification of the image specified with the IMAGE= statement.

CENTER
specifies center justification.

Alias: C

DEC
specifies aligning the values by the decimal point.

Alias: D

Restriction: Decimal alignment is supported for the printer family and RTF destinations only.

LEFT
specifies left justification.

Alias: L

RIGHT
specifies right justification.

Alias: R

Restriction: Not all contexts support RIGHT. If RIGHT is not supported, it is interpreted as CENTER.

Alias: JUST=

Restriction The TEXTALIGN= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: For the printer family destinations and the MARKUP destination, use the style attribute TEXTALIGN= with the style attribute VERTICALALIGN= in the style element PAGENO to control the placement of page numbers.

For example, this statement would produce a page number that is centered at the bottom of the page:

```
style PageNo from TitleAndFooters / textalign=c verticalalign=b;
```

Tip: For printer family destinations and the MARKUP destination, control the placement of dates by using the style attribute TEXTALIGN= with the style attribute VERTICALALIGN= in any of these style elements:

BODYDATE

DATE

For example, this statement would produce a date in the body file that is left justified at the top of the page:

```
style BodyDate from Date / textalign=l verticalalign=t;
```

TEXTDECORATION= BLINK | LINE_THROUGH | OVERLINE | UNDERLINE
changes the visual presentation of the text.

BLINK
specifies that the text's visual presentation alternates rapidly between visible and invisible.

Restriction: TEXTDECORATION=BLINK is valid only in the HTML and RTF destinations.

LINE_THROUGH
specifies that a line is drawn through the text.

Restriction: TEXTDECORATION=LINE_THROUGH is valid only in the HTML destination, the printer family, the measured RTF destination, and the RTF destinations.

OVERLINE
specifies that a line is drawn above the text.

Restriction: TEXTDECORATION=OVERLINE is valid only in the HTML destination and the printer family destinations.

UNDERLINE
specifies that a line is drawn below the text.

Restriction: TEXTDECORATION=UNDERLINE is valid only in the HTML destination, the printer family destinations, the measured RTF destination, and the RTF destination.

Tip: TEXTDECORATION= can be used with inline formatting and the ODS PDF statement to enhance PDF files.

TEXTINDENT=*n*
specifies the number of spaces that the first line of output will be indented.

Default: The default value for XML is 2. For all other ODS destinations, the default value is 0.

Alias: INDENT=

Restriction: The TEXTINDENT= attribute is valid only in the markup family destinations, the printer family destinations, and the RTF destination.

n
specifies the number of spaces to indent the output.

TICKDISPLAY= "INSIDE" | "OUTSIDE" | "ACROSS"
specifies the placement of all major and minor axix tick marks.

TEXTJUSTIFY= INTER_WORD | INTER_CHARACTER
specifies how to evenly distribute text.

INTER_WORD
specifies that the words will be evenly distributed across the page.

INTER_CHARACTER
specifies that all characters will be evenly distributed across a page.

Tip Use the TEXTJUSTIFY= style attribute with the TEXTALIGN=J (alias JUST=) style attribute.

TRANSPARENCY=*dimension*
specifies a transparency level for graphs. The values are 0.0 (opaque) to 1.0 (transparent).

Restriction: The TRANSPARENCY= attribute is valid only in the HTML destination.

See: dimension on page 535

URL="*uniform-resource-locator*"
specifies a URL to link to from the current cell.

Requirement: *uniform-resource-locator* must be enclosed in quotation marks.

Restriction: The URL= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

VERTICALALIGN= BOTTOM | MIDDLE | TOP
specifies vertical justification for graphs and cells. In graphs, this option specifies the vertical justification of the image specified with IMAGE=.

BOTTOM
specifies bottom justification.

Alias: B

MIDDLE
specifies center justification.

Alias: M

TOP
specifies top justification.

Alias: T

Alias: VJUST=

Restriction: The VERTICALALIGN= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip: For printer and markup family destinations, use the style attribute VERTICALALIGN= with the style attribute TEXTALIGN= in the style element PAGENO to control the placement of page numbers.

For example, this statement produces a page number that is centered at the bottom of the page:

```
style PageNo from TitleAndFooters / textalign=c verticalalign=b;
```

Tip: For printer and markup family destinations, control the placement of dates by using the style attribute VERTICALALIGN= with the style attribute TEXTALIGN= in any of these style elements:

BODYDATE

DATE

.

For example, this statement produces a date in the body file that is left justified at the top of the page:

```
style BodyDate from Date / textalign=l verticalalign=t;
```

VISITEDLINKCOLOR= *color*
specifies the color for links that have been visited in an HTML document.

Restriction: The VISITEDLINKCOLOR= attribute is valid only in markup family destinations.

See: color on page 534

WATERMARK= ON | OFF
specifies whether to make the image that is specified by BACKGROUNDIMAGE= into a watermark. A watermark appears in a fixed position as the window is scrolled.

ON
specifies to make the image that is specified by BACKGROUNDIMAGE= into a watermark.

OFF
specifies not to make the image that is specified by BACKGROUNDIMAGE= into a watermark.

Restriction: The WATERMARK= attribute is valid only in markup family destinations.

See also: BACKGROUNDIMAGE= on page 510

WIDTH= *dimension*
specifies the width of a cell, table, line, or a graph.

dimension
is a nonnegative number.

See: dimension on page 535

Alias: CELLWIDTH=

Alias: OUTPUTWIDTH=

Restriction: The HEIGHT= option does not apply to output generated as a result of GRSEG (graph segment) output.

Interaction: The XPIXELS= option in the SAS/GRAPH GOPTIONS statement overrides the WIDTH= attribute.

Tip: When used with graphs, the HEIGHT=*dimension* must be specified as a pixel or percentage value. If a unit of measure is not specified with the *dimension*, then the value will be in pixels. If a unit of measure other than pixels or percentage is specified with the *dimension*, then the HEIGHT=*dimension* is not applied to the graph.

Tip: A column of cells will have the width of the widest cell in the column.

Tip: Use WIDTH=100% to make the table or graph as wide as the window that it is open in.

Restriction: The WIDTH= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Style Attribute Values

Values for style attributes are one of the following:

color
is a string that identifies a color. A color is defined in the following ways:

- most of the color names that are supported by SAS/GRAPH. These names include the following:
  - a predefined SAS color (for example, blue or VIYG)
  - a red/green/blue (RGB) value (for example, CX0023FF)
  - a hue/light/saturation (HLS) value (for example, H14E162D)
  - a gray-scale value (for example, GRAYBB).
- an RGB value with a leading pound sign (#) rather than CX (for example, #0023FF).
- one of the colors that exists in the SAS session when the style is used:
  - DMSBLUE
  - DMSRED
  - DMSPINK
  - DMSGREEN
  - DMSCYAN
  - DMSYELLOW
  - DMSWHITE
  - DMSORANGE
  - DMSBLACK
  - DMSMAGENTA
  - DMSGRAY
  - DMSBROWN
  - SYSBACK
  - SYSSECB
  - SYSFORE

  Note: Use these colors only when running SAS in the windowing environment. △
- an English description of an HLS. Such descriptions use a combination of words to describe the lightness, the saturation, and the hue (in that order). Use the Color Naming System to form a color in the following ways:
  - combining a chromatic hue with a lightness, a saturation, or both
  - combining the achromatic hue gray with a lightness
  - combining the achromatic hue black or white without qualifiers

  Use the words in the following table:

Table 11.7 Hue/Light/Saturation (HLS) Values

Lightness	Saturation	Chromatic Hue	Achromatic Hue
		Blue	Black *
Very dark	Grayish	Purple	
Dark	Moderate	Red	

Lightness	Saturation	Chromatic Hue	Achromatic Hue
Medium	Strong	Orange | brown	Gray **
Light	Vivid	Yellow	
Very light		Green	
			White *

* Black and white cannot be combined with a lightness or a saturation value.
** Gray cannot be combined with a saturation value.

Combine these words to form a wide variety of colors. Here are examples:

- light vivid green
- dark vivid orange
- light yellow

Note: The Output Delivery System first tries to match a color with a SAS/GRAPH color. Thus, although brown and orange are interchangeable in the table, if you use them as unmodified hues, then they are different. The reason for this is that ODS interprets them as SAS colors, which are mapped to different colors. △

You can also specify hues that are intermediate between two neighboring colors. To do so, combine one of these adjectives with one of its neighboring colors:

- reddish
- orangish
- brownish
- yellowish
- greenish
- bluish
- purplish

For example, you can use the following as hues:

- bluish purple
- reddish orange
- yellowish green

See also: RBG Color Codes, HLS Color Codes, and Gray-Scale Color codes in *SAS/GRAPH: Reference* for information about SAS/GRAPH colors.

dimension

is a whole number, a percentage, or a nonnegative number followed by one of these units of measure:

Table 11.8 Units of Measure for Dimension

cm	Centimeters
em	Standard typesetting measurement unit for width
ex	Standard typesetting measurement unit for height
in	Inches

mm	Millimeters
pt	A printer's point

Default: For the PRINTER destination, units of 1/150 of an inch

font-definition
is the name of a font, the font size, and font keywords. A font definition has this general format:

("*font-face-1 <… , font-face-n>*", *font-size*, *keyword-list*)

"*font-face*"
specifies the name of the font.

ODS styles can now use new TrueType fonts. All Universal Printers and many SAS/GRAPH devices use the FreeType library to render TrueType fonts for output in all of the operating environments that SAS software supports. In addition, by default, many SAS/GRAPH device drivers and all Universal Printers generate output using ODS styles, and these ODS styles use TrueType fonts. In addition to SAS Monospace and SAS Monospace Bold, 21 new TrueType fonts are made available when you install SAS:

- five Latin fonts compatible with Microsoft
- eight multilingual Unicode fonts
- eight monolingual Asian fonts

For more information about the TrueType fonts, see the section "Printing with SAS" in *SAS Language Reference: Concepts*.

Restriction: You must enclose multiple *font-face* in quotation marks. If you specify only one font and if its name does not include a space character, then omit the quotation marks.

Tip: If you specify more than one font, then the destination device uses the first one that is installed on the system.

font-size
specifies the size of the font. *font-size* is a dimension or a number without units of measure. If you specify a dimension, then specify a unit of measure. Without a unit of measure the number becomes a size that is relative to all other font sizes in the HTML document. For more information, see dimension on page 535.

keyword-list
specifies the font weight, font style, and font width. Include one value for each, in any order. This table shows the keywords to use:

Table 11.9 Font Keywords

Keywords for Font Weight	Keywords for Font Style	Keywords for Font Width
MEDIUM	ITALIC	NORMAL*
BOLD	ROMAN	COMPRESSED*
DEMI_BOLD*	SLANT	EXTRA_COMPRESSED*
EXTRA_BOLD*		NARROW*
LIGHT		WIDE*

Keywords for Font Weight	Keywords for Font Style	Keywords for Font Width
DEMI_LIGHT*		EXPANDED*
EXTRA_LIGHT*		

* Few fonts honor these values.

Featured in: Example 2 on page 557

format
is a SAS format or a user-defined format.

integer | integer-list | integer-column-list
specifies a column variable that contains integer values, or a dynamic variable that refers to such a column variable.

integer
specifies a single integer.

integer-list
specifies a sequence of integer values, or a column variable that contains integer values, or a dynamic variable that refers to such a column variable or to a string.

integer-column-list
specifies a sequence of column variables, or a column variable that contains column variables, or a dynamic variable that refers to such a column variable, or a dynamic variable that refers to a string containing a list of column variables. Values within the columns must be integers.

style-reference
is a reference to an attribute that is defined in the current style or in the parent style (or beyond). The value used is the name of the style element followed by the name of an attribute, in parentheses, within that element. Style references have the following form:

style-attribute=target-style-element("target-style-attribute")

style-attribute	specifies the name of the style attribute.
target-style-element	specifies the name of the style element that contains the style attribute that you want to reference.
target-style-attribute	specifies the style attribute with the value that you want to use.

Requirement: You must enclose *target-style-attribute* in quotation marks if it is a user-supplied style attribute.

Featured in: Example 2 on page 557

See also: "Understanding Style References" on page 544

"*string*"
is a quoted character string.

user-defined-format
specifies a format created with the FORMAT procedure.

Restriction *user-defined-format* can only be specified for data cells.

Concepts: Styles and the TEMPLATE Procedure

Viewing the Contents of a Style

To view the contents of a style, use the SAS windowing environment, the command line, or the TEMPLATE procedure.

- Using the SAS Windowing Environment
  1. In the Results window, select the **Results** folder. Right-click and select **Templates** to open the Templates window.
  2. Double-click **Sashelp.Tmplmst** to view the contents of that directory.
  3. Double-click **Styles** to view the contents of that directory.
- Using the Command Line
  1. To view the Templates window, submit this command in the command line:

     ```
     odstemplates
     ```

     The Templates window contains the item stores **Sasuser.Templat** and **Sashelp.Tmplmst**.
  2. Double-click an item store, such as **Sashelp.Tmplmst**, to expand the list of directories where ODS templates are stored. The templates that SAS provides are in the item store Sashelp.Tmplmst.
  3. To view the styles that SAS provides, double-click the **Styles** item store.
  4. Right-click the style, such as **Journal**, and select **Open**. The style template is displayed in the Template Browser window.
- Using the TEMPLATE Procedure
  1. Submit this code to view the contents of the default HTML style that SAS supplies.

     ```
     proc template;
     source styles.default;
     run;
     ```

  2. View any of the SAS styles by specifying the styles.*style-template* in the SOURCE statement. The SAS styles are in the Sashelp.Tmplmst item store.

Working with Styles

Finding and Viewing the Default Style for ODS Destinations

The default styles for the ODS output destinations are stored in the STYLES directory in the template store Sashelp.Tmplmst, along with the other styles that are supplied by SAS. You can view the styles from the TEMPLATE window, or you can submit this PROC TEMPLATE step to write the style to the SAS log:

```
proc template;
   source styles.template-name;
run;
```

The following table lists the ODS destinations and their default styles:

Table 11.10 Destination Category Table

Destinations	Default Style Name
LISTING	Listing
HTML	Default
MARKUP Language Tagsets	Default
PRINTER	Printer for PDF and PS, monochromePrinter for PCL
RTF	RTF

Modifying Style Elements in the Default Style for HTML and Markup Languages

When you work with styles, it is often more efficient to modify a SAS style than to write a completely new style. Example 3 on page 564 shows you how to modify the default style.

To customize the style for use at a specific site, it is helpful to know what each style element in the style specifies. For a list of the default HTML and markup languages style elements, see Appendix 4, "ODS Style Elements," on page 905.

ODS Styles with Graphical Style Information

SAS provides ODS styles that incorporate graph style information. These styles utilize a number of style attributes that are used by other style elements, but they also use several style attributes that are unique to graph styles. For example, use the STARTCOLOR= style attribute and the ENDCOLOR= style attribute to produce a gradient effect that gradually changes from the starting color to the ending color in a specified element. When either the STARTCOLOR= style attribute or the ENDCOLOR= style attribute, but not both, is specified, then the style attribute that was not specified is transparent when the TRANSPARENCY= style attribute is being used. In Example 4 on page 570, only the ENDCOLOR= style attribute is specified. Therefore, the starting color is transparent.

The TRANSPARENCY= style attribute is another style attribute that is unique to graph styles. With transparency, specify the level of transparency (from 0.0 to 1.0) to indicate the percentage of transparency (0 to 100 %) for the graph element. While you can use the BACKGROUNDIMAGE= style attribute in other style elements to stretch an image, in graph styles, you can also use the IMAGE= style attribute to position or tile an image.

With graph styles, elements, or templates you can also combine images and colors to create a blending affect. The blending works best when you use a grayscale image with a specified color. Blending is done in these style elements: GraphLegendBackground, GraphCharts, GraphData#, GraphFloor, and GraphWalls. To blend, specify a color using the BACKGROUNDCOLOR= or COLOR= style attribute and specify an image using the BACKGROUNDIMAGE= or IMAGE= style attribute.

Note: When using the GraphData# style element, you can use the COLOR= style attribute, but not the BACKGROUNDCOLOR= style attribute to specify a color value. △

See “Style Attributes and Their Values” on page 498 for a complete listing of style attributes. For a complete list of style elements see Appendix 4, “ODS Style Elements,” on page 905.

In addition to using defined ODS styles, you can also modify an existing style or create an entirely new style using the new graph style elements. Example 4 on page 570 describes how a defined ODS style was generated.

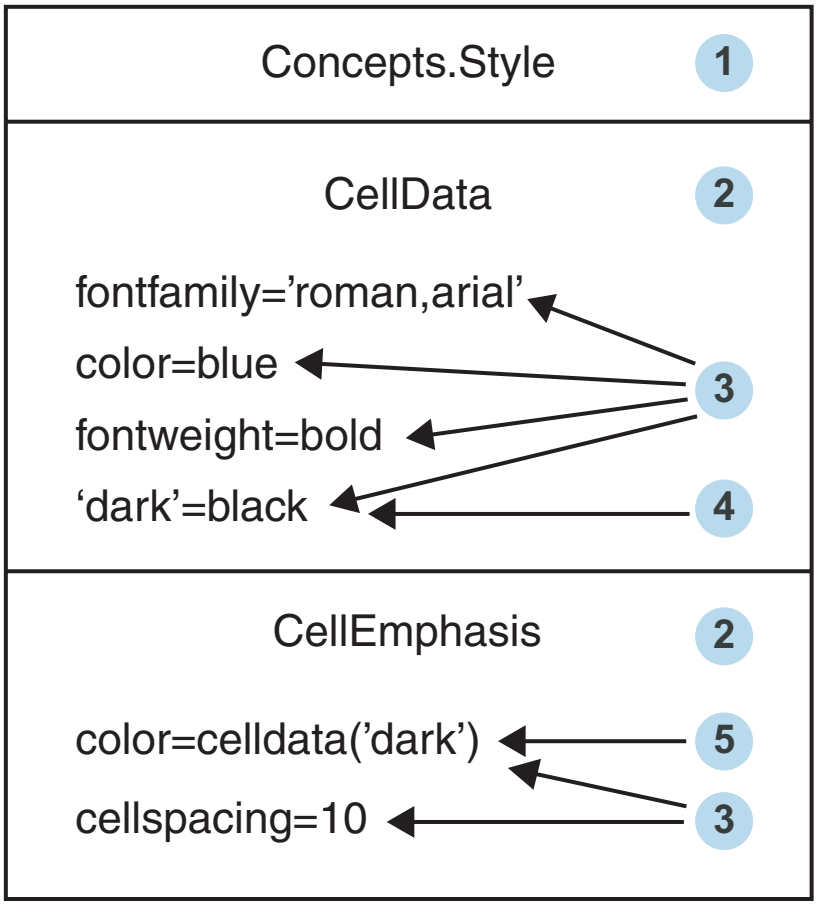
See “Viewing the Contents of a Style” on page 538 for information about viewing the code for the ODS styles that are delivered with SAS.

Understanding Styles, Style Elements, and Style Attributes

To help you become familiar with styles, style elements, and style attributes, look at the relationship between them. The following program creates a style, Concepts.Style. The diagram that follows the program shows the relationship between the style, the style elements, and the style attributes.

```
proc template;
   define style concepts.style;
      style celldata /
         fontfamily="roman, arial"
         color=blue
         fontweight =bold
         "dark"=black;
      style cellemphasis from celldata /
         color=celldata("dark")
         cellspacing=10;
   end;
run;
```

Display 11.5 Diagram of a Style, Including Style Elements and Style Attributes



The following list corresponds to the numbered items in the preceding diagram:

❶ Concepts.Style is a *style*. Styles describe how to display presentation aspects (color, font, font size, and so on) of the output for an entire SAS job. A style determines the overall appearance of the ODS documents that use it. Each style is composed of style elements. Styles are created with the "DEFINE STYLE Statement" on page 490.

New styles can be created independently or from an existing style. You can use the "PARENT= Statement" on page 494 to create a new style from an existing style.

❷ CellData and CellEmphasis are *style elements*. A style element is a collection of style attributes that apply to a particular part of the output for a SAS program. For example, a style element might contain instructions for the presentation of column headings or for the presentation of the data inside table cells. Style elements might also specify default colors and fonts for output that uses the style. Style elements exist inside of styles and are defined by the "STYLE Statement" on page 495.

Note: For a list of the default style elements used for HTML and markup languages and their inheritance, see Appendix 4, "ODS Style Elements," on page 905. △

❸ The following are *style attribute-value pairs*:

- **fontfamily="roman, arial"**
- **color=blue**
- **fontweight=bold**
- **"dark"=black**
- **color=celldata("dark")**
- **cellspacing=10**

Style attributes specify a value for one aspect of the presentation. For example, the COLOR= attribute specifies the value **blue** for the foreground color of a table, and the FONTFAMILY= attribute specifies the values **roman** and **arial** as the font to use.

Style attributes exist within style elements and can be supplied by SAS or be user-defined. FONTFAMILY=, COLOR=, FONTWEIGHT=, and CELLSPACING= are style attributes supplied by SAS. For a list of style attributes supplied by SAS, see “Style Attributes and Their Values” on page 498.

❹ "Dark" is a user-defined style attribute. It specifies to substitute the value **black** whenever the value **"dark"** is specified.

❺ The value **celldata("dark")** is a style reference. Style attributes can be referenced with style references. This style reference specifies that PROC TEMPLATE go to the CellData style element and use the value that is specified for the "dark" style attribute. See the *style-reference* value in the section “Style Attribute Values” on page 534 for more information about style references.

Understanding Inheritance

Overview

Inheritance can be initiated by the PARENT= statement or the FROM= option in the STYLE statement.

The PARENT= statement specifies that PROC TEMPLATE copy all of the style elements from the parent style to the new child style. The style elements are used in the new template unless the new template has style elements that overrides them.

The FROM= option specifies that PROC TEMPLATE copy all of the style attributes from the parent style element to the specified child style element.

Inheritance Between Styles

Inheritance between styles is initiated by the PARENT= option, and involves the following process:

1. When the PARENT= statement is specified, style elements in the parent style are copied into the new style. This copying occurs before any inheritance can occur within the new style.
2. If there is a like-named style element within the child style that *does not* have a FROM option specified, then the style element from the child style overrides the style element from the parent style.
3. If there is a like-named style element within the child style that *does* have the FROM option specified, then the child style element absorbs the style attributes from the parent style element. If there are like-named style attributes in the two style elements, then the style attributes from the child style element are used.

The following code shows an example of inheritance between two styles:

Example Code 11.1 Original Code for Creating Style2

```
define style style1;
    style fonts /
        "docfont" = ("Arial", 3)
        "tablefont" = ("Times", 2);
     style output /
        cellpadding = 5
        cellspacing = 0
        font = fonts("docfont");
    style table from output /
        cellspacing = 2
        font = fonts("tablefont");
     style header /
         backgroundcolor=white
         color=blue
         fontfamily="arial, helvetica"
         fontweight=bold;
 end;

define style style2;
   parent = style1;
     style fonts from fonts /
```

```
        "docfont" = ("Helvetica", 3);
     style table from table /
        cellspacing = 4;
     style header /
         fontstyle=roman
         fontsize=5;
end;
```

The Style2 style from the previous code could also be written this way:

Example Code 11.2 Expanded Version of Style2

```
define style style2;
   style fonts/
      "docfont" = ("Helvetica", 3)
      "tablefont" = ("Times", 2);
   style output /
      cellpadding = 5
      cellspacing = 0
      font = fonts("docfont");
   style table from output /
      cellspacing=4
      font = fonts("tablefont");
   style header /
         fontstyle=roman
         fontsize=5;
end;
```

Inheritance Between Style Elements

The FROM option on a STYLE statement is used to initiate inheritance from another style element. The style element referenced by the FROM option can exist in either the current style or the parent style (if a parent template is specified using the PARENT= statement).

For example, in both the original Style2 style and the expanded version from the section "Inheritance Between Styles" on page 543 the Table style element, which is created with the **`style table from output / ...`** statement, ends up with the following style attributes:

- **`cellpadding= 5`**
- **`cellspacing= 4`**
- **`font=fonts("tablefont")`**

Understanding Style References

A style reference references a style attribute in a style element. The style element can exist either in the current style or in the parent style.

For example, suppose that you create a style element named DataCell that uses the COLOR= and BACKGROUNDCOLOR= style attributes:

```
style datacell / backgroundcolor=blue
                 color=white;
```

To ensure that another style element, NewCell, uses the same background color, use a style reference in the NewCell element, like this:

```
style newcell / backgroundcolor=datacell(backgroundcolor);
```

The style reference **datacell(backgroundcolor)** indicates that the value for the style attribute BACKGROUNDCOLOR= of the style element named DataCell should be used.

Similarly, suppose that you create a style element named HighLighting that defines three style attributes:

```
style highlighting /
   "go"=green
   "caution"=yellow
   "stop"=red;
```

You can then define a style element named Messages that references the colors that are defined in the HighLighting style element:

```
style messages;
   "note"=highlighting("go")
   "warning"=highlighting("caution")
   "error"=highlighting("stop");
```

Because you used style references, multiple style elements can use the colors defined in the HighLighting style element. If you change the value of **go** to blue in the HighLighting style element, then every style element that uses the style reference **highlighting("go")** will use blue instead of green.

In the following code, the FONT= style attribute in the Output style element is defined in terms of the Fonts style element. The value **fonts("docfont")** tells PROC TEMPLATE to go to the last instance of the style element named Fonts and use the value for the style attribute DocFont.

The FONT= style attribute in the Table style element is also defined in terms of the Fonts style element. The value **fonts("tablefont")** tells PROC TEMPLATE to go to the last instance of the style element named Fonts and use the value for the style attribute TableFont.

Example Code 11.3 Program with Unresolved Style References

```
define style style1;
    style fonts /
        "docfont" = ("Arial", 3)
        "tablefont" = ("Times", 2);
     style output /
        cellpadding = 5
        cellspacing = 0
        font = fonts("docfont");
    style table from output /
        cellspacing = 2
        font = fonts("tablefont");
     style header /
         backgroundcolor=white
         color=blue
         fontfamily="arial, helvetica"
         fontweight=bold;
 end;

define style style2;
   parent = style1;
     style fonts from fonts /
        "docfont" = ("Helvetica", 3);
     style table from table /
        cellspacing = 4;
```

```
     style header /
         fontstyle=roman
         fontsize=5;
end;
```

When you submit the code in SAS, the output is created as if you submitted the following program. Notice that in the Output style element, the style reference resolves to **("helvetica", 3)**, not **("Arial", 3)**. This is because the "DocFont" user-supplied style attribute in the Style2 style overrides the like-named style attribute in the Style1 style.

Example Code 11.4 Program with Resolved Style References

```
define style style1;
    style fonts /
        "docfont" = ("Arial", 3)
        "tablefont" = ("Times", 2);
     style output /
        cellpadding = 5
        cellspacing = 0
   /*** Resolved from "docfont" in Style2***/
        font = fonts("helvetica", 3);
    style table from output /
        cellspacing = 2
   /*** Resolved from "tablefont" in Style1***/
        font = fonts("Times", 2);
     style header /
         backgroundcolor=white
         color=blue
         fontfamily="arial, helvetica"
         fontweight=bold;
 end;

define style style2;
   parent = style1;
     style fonts from fonts /
        "docfont" = ("Helvetica", 3);
     style table from table /
        cellspacing = 4;
     style header /
         fontstyle=roman
         fontsize=5;
end;
```

Using the FROM Option

The FROM option is used with a style element in order to inherit from another style element. This can result in an incomplete style element in the child style.

For example, in Program 1 the style Concepts.Style2 inherits all of its style elements and style attributes from the style Concepts.Style1. However, the instance of the style element Colors in Concepts.Style2 overrides the instance of Colors in Concepts.Style1. This is because there is no FROM option in the STYLE statement that creates Colors in Concepts.Style2. Therefore, Colors has one style attribute: **"dark"=dark blue**.

When you run the program, the only style references to Color that resolve are references that refer to the **"dark"** style attribute. Style references in Concepts.Style1

and Concepts.Style2 such as **colors("fancy")** and **colors("medium")** do not resolve because they refer to attributes that were not copied into the current instance of the Colors style element. The resulting output is Display 11.6 on page 548.

To correct this, you can add the FROM option to the STYLE statement that creates the Colors style element in Concepts.Style2:

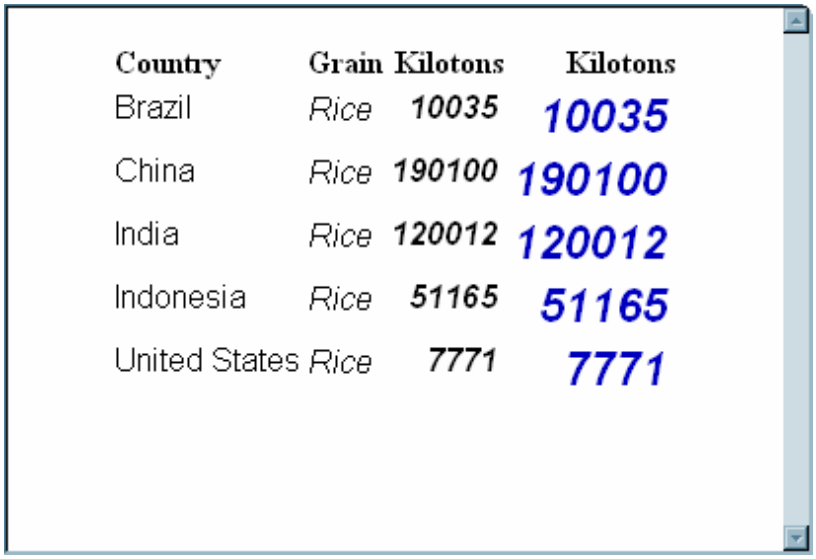
```
style colors from colors /
   "dark"=dark blue;
```

Example Code 11.5 Program 1: Creating the Colors Style Element without the FROM Option

```
proc template;
   define style concepts.style1;
      style colors /
         "default"=white
         "fancy"=very light vivid blue
         "medium"=red ;
      style celldatasimple /
         fontfamily=arial
         backgroundcolor=colors("fancy")
         color=colors("default");
      style celldataemphasis from celldatasimple /
         color=colors("medium")
         fontstyle=italic;
      style celldatalarge from celldataemphasis /
         fontweight=bold
         fontsize=3;
   end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
      style colors /
         "dark"=dark blue;
      style celldataemphasis from celldataemphasis /
         backgroundcolor=white;
      style celldatasmall from celldatalarge /
         fontsize=5
         color=colors("dark")
         backgroundcolor=colors("medium");
   end;
run;
```

For the complete SAS code that created the following output, see the version of the code without the FROM option in “Using the FROM option” on page 883.

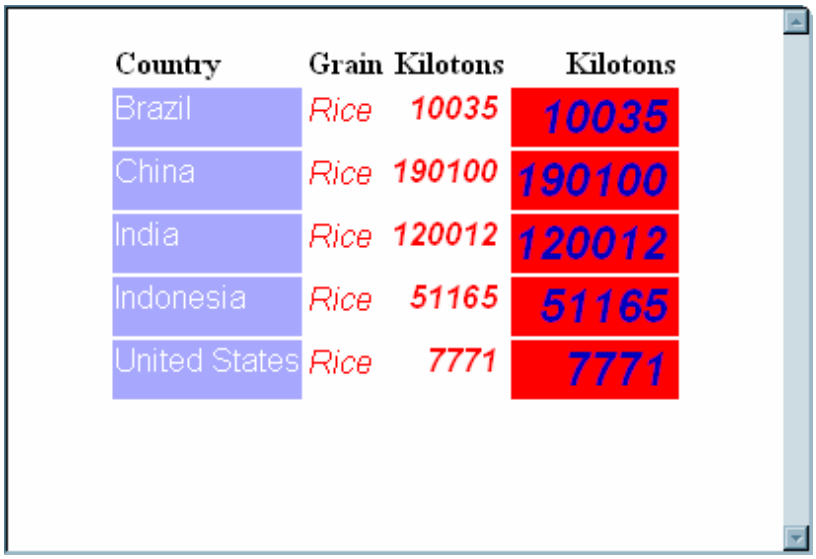
Display 11.6 Output Created without the FROM Option



Country	Grain	Kilotons	Kilotons
Brazil	Rice	10035	10035
China	Rice	190100	190100
India	Rice	120012	120012
Indonesia	Rice	51165	51165
United States	Rice	7771	7771

For the complete SAS code that created the following output, see the version of the code with the FROM option in “Using the FROM option” on page 883.

Display 11.7 Output Created with the FROM Option



Country	Grain	Kilotons	Kilotons
Brazil	Rice	10035	10035
China	Rice	190100	190100
India	Rice	120012	120012
Indonesia	Rice	51165	51165
United States	Rice	7771	7771

Inheritance Compatibility across Versions

In most cases, an ODS style element or style that was created in a previous version of SAS will still be compatible with later versions of SAS. However, beginning with SAS 9.2, style inheritance is completely expanded before style element inheritance takes place. This change can cause discrepancies between the output a program creates in a previous version of SAS and the output that same program creates in SAS 9.2.

The following program creates different output depending on whether it is run in SAS 9.2 or in a previous version of SAS. In SAS 9.2, the yellow background that

CellDataEmphasis has in Concepts.Style2 is passed to CellDataLarge and CellDataSmall. However, in previous versions of SAS, the yellow background is not passed to CellDataLarge and CellDataSmall. For more information about the using the FROM option, see "Using the FROM Option" on page 546.

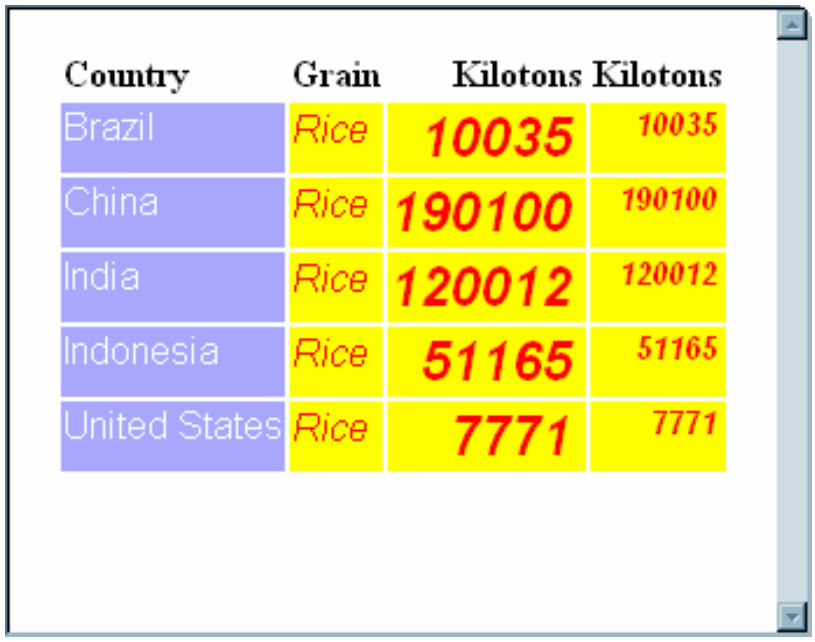
Example Code 11.6 Program 2: Using Inheritance to Create a New Style Element from a Style Element in the Parent Style

```
proc template;
   define style concepts.style1;
      style celldatasimple /
         fontfamily=arial
         backgroundcolor=very light vivid blue
         color=white;
      style celldataemphasis from celldatasimple /
         color=red ❶
         fontstyle=italic;
      style celldatalarge from celldataemphasis /
         fontweight=bold
         fontsize=5;
   end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
      style celldataemphasis from celldataemphasis/ ❸
         backgroundcolor=yellow; ❷
      style celldatasmall from celldatalarge /
         fontsize=2;
   end;
run;
```

The output this program creates when you run it in a previous version of SAS is different from the output the program creates in SAS 9.2. This is because, when you change the value of the COLOR= attribute in CellDataEmphasis from red (❶) to yellow (❷), the change affects only style elements that inherit from CellDataEmphasis *in Concepts.Style2* (❸). Within Concepts.Style2, there are no style elements that inherit from CellDataEmphasis. Therefore, only CellDataEmphasis in Concepts.Style2 has yellow text. Beginning with SAS 9.2, all style elements in parent style definitions also pick up the color change.

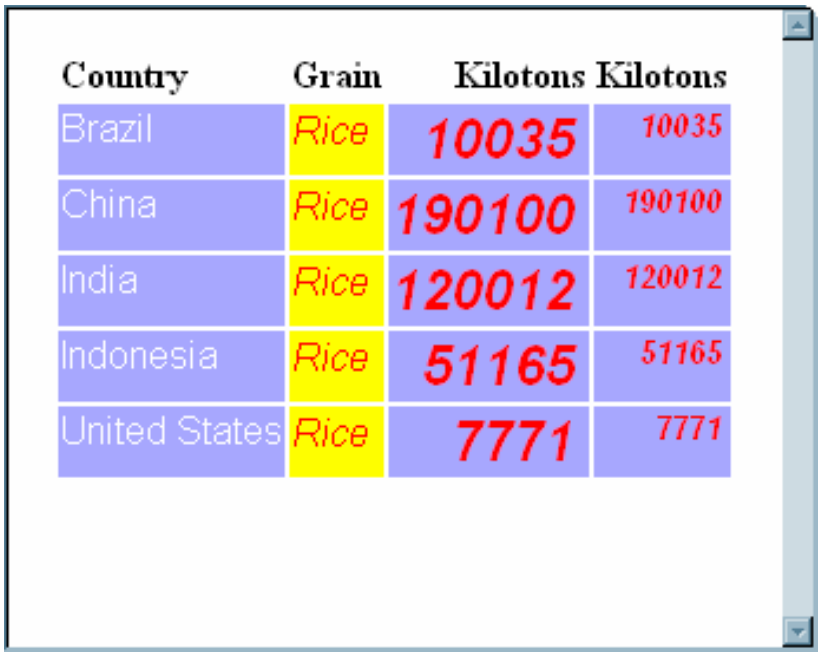
For the complete SAS code that created this output, see the SAS 9.1 version of the code in "Inheritance Compatibility Across SAS Versions" on page 886.

Display 11.8 SAS 9.2 Output



Country	Grain	Kilotons	Kilotons
Brazil	Rice	10035	10035
China	Rice	190100	190100
India	Rice	120012	120012
Indonesia	Rice	51165	51165
United States	Rice	7771	7771

Display 11.9 SAS 9.1 Output



Country	Grain	Kilotons	Kilotons
Brazil	Rice	10035	10035
China	Rice	190100	190100
India	Rice	120012	120012
Indonesia	Rice	51165	51165
United States	Rice	7771	7771

Examples: Creating and Modifying Styles Using the TEMPLATE Procedure

Example 1: Creating a Stand-Alone Style

PROC TEMPLATE features:

DEFINE STYLE statement:
- STYLE statement:
  - BACKGROUNDCOLOR=
  - BORDERWIDTH=
  - CELLSPACING=
  - FONTFAMILY=
  - FONTSIZE=
  - FONTSTYLE=
  - FONTWEIGHT=
  - COLOR=

DEFINE TABLE statement:
- CLASSLEVELS= table attribute
- DYNAMIC statement
- MVAR statement

DEFINE COLUMN statement:
- BLANK_DUPS=
- GENERIC=
- HEADER=
- STYLE=

DEFINE FOOTER statement:
- TEXT statement

Other ODS features:

ODS HTML statement

ODS LISTING statement

FILE statement with ODS= option

PUT statement with _ODS_ argument

Data set: See “Creating the Grain_Production Data Set” on page 878.

Format: See “Creating the $CNTRY Format” on page 869.

Program Description

This example creates a style that is not based on any other style. When you create a style, you will usually base it on one of the styles that SAS provides (see Example 3 on page 564). However, this example is provided to show you some of the basic ways to create a style.

It is important to understand that by default, certain table elements are created with certain style elements. For example, unless you specify a different style element with

the STYLE= attribute, ODS produces SAS titles with the SystemTitle style element. Similarly, unless you specify otherwise, ODS produces headers with the Header style element. (For information about each style element, see Appendix 4, "ODS Style Elements," on page 905.

Program

Create a new style named NewStyle with the style element CellContents. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called NewStyle. This STYLE statement defines the style element CellContents. This style element is composed of the style attributes that appear on the STYLE statement. The FONTFAMILY= attribute tells the browser to use the Arial font if it is available, and to look for the Helvetica font if Arial is not available.

```
proc template;
   define style newstyle;
      style cellcontents /
         fontfamily="arial, helvetica"
         fontweight=medium
         backgroundcolor=blue
         fontstyle=roman
         fontsize=5
         color=white
```

Create the style element Header. This STYLE statement creates the style element Header. By default, ODS uses Header to produce both spanning headers and column headings. This style element uses a different background color from CellContents. It uses the same font (Arial or Helvetica), the same font style (roman), the same font color (white), and the same font size (5) as CellContents.

```
      style header /
         backgroundcolor=very light blue
         fontfamily="arial, helvetica"
         fontweight=medium
         fontstyle=roman
         fontsize=5
         color=white;
```

Create the style element SystemTitle. This STYLE statement creates the style element SystemTitle. By default, ODS uses SystemTitle to produce SAS titles. This style element uses a color scheme of a red foreground on a white background. It uses the same font and font weight as Header and CellContents, but it adds an italic font style and uses a larger font size.

```
      style systemtitle /
         fontfamily="arial, helvetica"
         fontweight=medium
         backgroundcolor=white
         fontstyle=italic
         fontsize=6
         color=red;
```

Create the style element Footer. This STYLE statement creates the style element Footer. This style element inherits all the attributes of SystemTitle. However, the font size that it inherits is overwritten by the FONTSIZE= attribute in its template.

```
      style footer from systemtitle /
          fontsize=3;
```

Create the style element Table. This STYLE statement creates the style element Table. By default, ODS uses this style element to display tables.

```
      style table /
          cellspacing=5
          borderwidth=10;
```

End the style. The END statement ends the style template. The RUN statement executes the TEMPLATE procedure.

```
    end;
run;
```

Create the table template Table1. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE TABLE statement creates a new table template called Table1.

```
proc template;
   define table table1;
```

Specify the symbol that references one macro variable. The MVAR statement defines a symbol, SysDate9, that references a macro variable. ODS will use the value of this macro variable as a string. References to the macro variable are resolved when ODS binds the table template to the data component to produce an output object. SYSDATE9 is an automatic macro variable whose value is always available.

```
    mvar sysdate9;
```

Specify the symbol that references a value to be supplied by the data component. The DYNAMIC statement defines a symbol, Colhd, that references a value that the data component supplies when ODS binds the template and the data component to produce an output object. The values for Colhd are provided in the FILE statement in the DATA step that appears later in the program. Using dynamic column headings gives you more flexibility than does hard-coding the headers in the table template.

```
    dynamic colhd;
```

Control the repetition of values that do not change from one row to the next row. The CLASSLEVELS= attribute suppresses the display of the value in a column that is marked with BLANK_DUPS=ON if the value changes in a previous column that is also marked with BLANK_DUPS=ON. Because BLANK_DUPS= is set in a generic column, set this attribute as well.

```
    classlevels=on;
```

Create the column Char_Var. This DEFINE statement and its attributes create the column template Char_Var.

GENERIC= specifies that multiple variables can use the same column template.

BLANK_DUPS= suppresses the display of the value in the column if it does not change from one row to the next (and, because CLASSLEVELS=ON for the table, if no values in preceding columns that are marked with BLANK_DUPS=ON changes).

HEADER= specifies that the header for the column will be the text of the dynamic variable Colhd, whose value will be set by the data component.

The STYLE= attribute specifies that the style element for this column template is CellContents.

The END statement ends the template.

```
    define column char_var;
       generic=on;
       blank_dups=on;
       header=colhd;
       style=cellcontents;
    end;
```

Create the column template Num_Var. This DEFINE statement and its attributes create the column template Num_Var. GENERIC= specifies that multiple variables can use the same column template. HEADER= specifies that the header for the column will be the text of the dynamic variable Colhd, whose value will be set by the data component.

The STYLE= attribute specifies that the style element for this column template is CellContents.

The END statement ends the template.

```
    define column num_var;
       generic=on;
       header=colhd;
       style=cellcontents;
    end;
```

Create the footer element Table_Footer. The DEFINE statement and its substatement define the table element Table_Footer. The FOOTER argument declares Table_Footer as a footer. The TEXT statement specifies the text of the footer. When ODS binds the data component to the table template (in the DATA step that follows), it will resolve the value of the macro variable SYSDATE9.

```
    define footer table_footer;
       text "Prepared on " sysdate9;
    end;
```

End the table template. This END statement ends the table template. The RUN statement executes the PROC TEMPLATE step.

```
    end;
run;
```

Stop the creation of the listing output. The ODS LISTING statement closes the LISTING destination in order to conserve resources. The LISTING destination is open by default.

```
ods listing close;
```

Create HTML output and specify the location for storing the HTML output. Specify the style to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file NewStyle-Body in the current directory. The STYLE= option tells ODS to use NewStyle as the style when it formats the output.

```
ods html body="newstyle-body.htm"
         style=newstyle;
```

Specify the titles for the report. The TITLE statements provide two titles for the output.

```
   title "Leading Grain Producers";
   title2 "in 1996";
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object.

The SET statement reads the data set Grain_Production. The WHERE statement subsets the data set so that the output object contains information only for rice and corn production in 1996.

```
data _null_;
   set grain_production;
   where type  in ("Rice", "Corn") and year=1996;
```

Route the DATA step results to ODS and use the Table1 table template. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, "Output Delivery System and the DATA Step," on page 39.) The TEMPLATE= suboption tells ODS to use the table template named Table1, which was previously created with PROC TEMPLATE.

```
   file print ods=(
        template="table1"
```

Specify the column template to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table template. For example, the first *column-specification* specifies that the first column of the output object contains the values of the variable COUNTRY and that it uses the column template named Char_Var. GENERIC= must be set to ON in both the table template and each column assignment in order for multiple variables to use the same column template. The FORMAT= suboption specifies a format for the column. The DYNAMIC= suboption provides the value of the dynamic variable Colhd for the current column. Notice that for the first column the column header is Country, and for the second column, which uses the same column template, the column header is Year.

```
        columns=(
           char_var=country(generic=on format=$cntry.
                    dynamic=(colhd="Country"))
           char_var=type(generic dynamic=(colhd="Year"))
           num_var=kilotons(generic=on format=comma12.
                   dynamic=(colhd="Kilotons"))
           )
        );
```

Write the data values to the data component. The _ODS_ option and the PUT statement write the data values for all columns to the data component. The RUN statement executes the DATA step.

```
    put _ods_;
run;
```

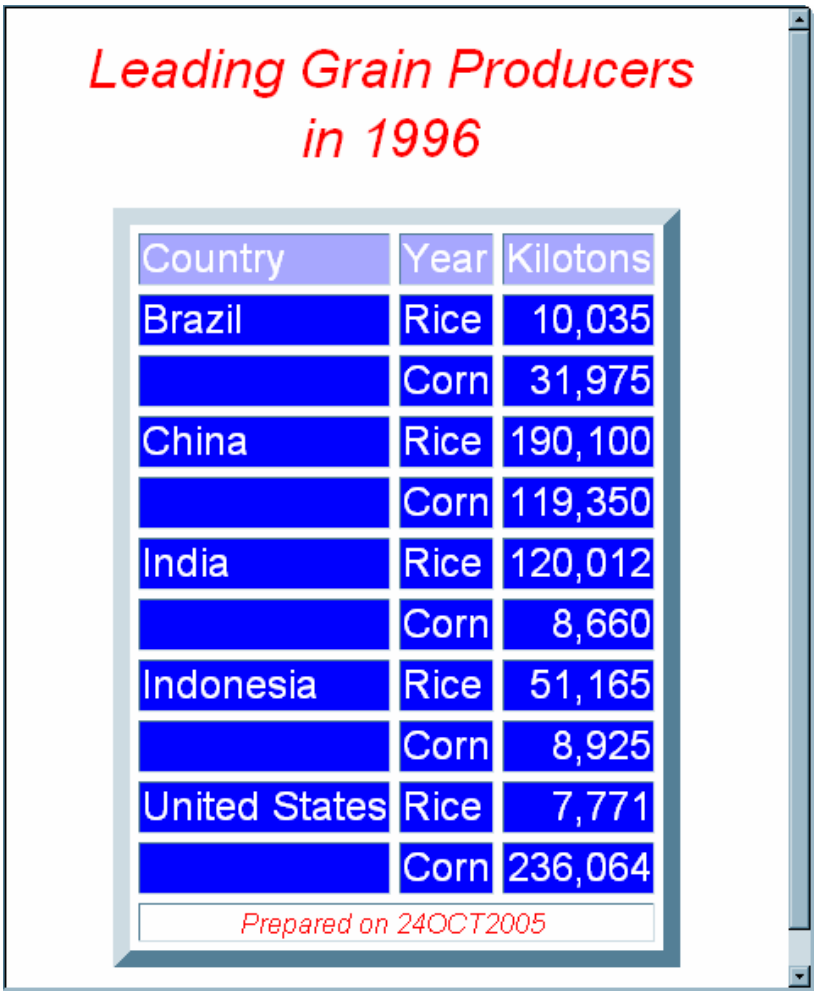
Stop the creation of the HTML output and create the listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. Close the destination so that you can view the output with a browser. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods html close;
ods listing;
```

HTML Output: Specifying Colors and Fonts with User-Defined Attributes

Display 11.10 HTML Output

Use the fonts to confirm that SAS titles use the SystemTitle style element, that column headings use the Header style element, that the footer uses the Table-Footer style element, and that the contents of both character and numeric cells use the CellContents style element. Use the width of the table border and the spacing between cells to confirm that the table itself is produced with the Table style element.



Leading Grain Producers in 1996

Country	Year	Kilotons
Brazil	Rice	10,035
	Corn	31,975
China	Rice	190,100
	Corn	119,350
India	Rice	120,012
	Corn	8,660
Indonesia	Rice	51,165
	Corn	8,925
United States	Rice	7,771
	Corn	236,064

Prepared on 24OCT2005

Example 2: Using User-Defined Attributes

PROC TEMPLATE features:

DEFINE STYLE statement:
- STYLE statement with user-defined attributes:
- DEFINE TABLE statement:
  - CLASSLEVELS= table attribute
  - DYNAMIC statement
  - MVAR statement
- DEFINE COLUMN statement:
  - BLANK_DUPS=
  - GENERIC=
  - HEADER=
  - STYLE=

DEFINE COLUMN statement:
- BLANK_DUPS= attribute
- CELLSTYLE-AS statement
- GENERIC= attribute

DEFINE FOOTER statement:
- TEXT statement

Other ODS features:

ODS HTML statement

ODS LISTING statement

FILE statement with ODS= option

PUT statement with _ODS_ argument

Data set: See "Creating the Grain_Production Data Set" on page 878.

Format: See "Creating the $CNTRY Format" on page 869.

Program 1: Description

This example creates a style that is equivalent to the style that Example 1 on page 551 creates. However, this style uses user-defined attributes to specify colors and fonts. This technique makes it possible to easily make changes in multiple places in the output.

Program 1: Creating the Style

Create the style NewStyle2. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called NewStyle2. This STYLE statement defines the style element Fonts.

This style element is composed of three user-defined attributes: CellFont, HeadingFont, and TitleFont. Each of these attributes describes a font. This style specifies the fontfamily, fontsize, fontweight, and the fontstyle for each of the three attributes. The font and fontwidth attributes are still defined by the default style.

```
proc template;
   define style newstyle2;
```

```
   style fonts /
      "cellfont"=("arial, helvetica", 4, medium roman)
      "headingfont"=("arial, helvetica", 5, bold roman)
      "titlefont"=("arial, helvetica", 6, bold italic);
```

Create the style element Colors. This STYLE statement defines the style element Colors. This style element is composed of four user-defined attributes: light, medium, dark, and bright. The values for medium and dark are RGB values equivalent to very light blue and blue.

```
   style colors /
      "light"=white
      "medium"=cxaaaaff
      "dark"=cx0000ff
      "bright"=red;
```

Create the three style elements CellContents, Header, and SystemTitle. Create the style element Footer using inheritance. The style attributes are defined in terms of the user-defined attributes that were created earlier in the style. For example, the foreground color in CellContents is set to **`colors("light")`**. Looking at the template of Colors, you can see that this is white. However, by setting the colors up in a style element with user-defined attributes, you can change the color of everything that uses a particular color by changing a single value in the style element Colors.

```
   style cellcontents /
      backgroundcolor=colors("dark")
      color=colors("light")
      font=fonts("cellfont");
   style header /
      backgroundcolor=colors("medium")
      color=colors("dark")
      font=fonts("headingfont");
   style systemtitle /
      backgroundcolor=colors("light")
      color=colors("bright")
      font=fonts("titlefont");
   style footer from systemtitle /
      fontsize=3;
   style table /
      cellspacing=5
      borderwidth=10;
```

End the style. The END statement ends the style. The RUN statement executes PROC TEMPLATE.

```
   end;
run;
```

Create the table template Table1. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE TABLE statement creates a new table template called Table1.

```
proc template;
   define table table1;
```

Specify the symbol that references one macro variable. The MVAR statement defines a symbol, Sysdate9, that references a macro variable. ODS will use the value of this macro variable as a string. References to the macro variable are resolved when ODS binds the table template to the data component to produce an output object. SYSDATE9 is an automatic macro variable whose value is always available.

```
mvar sysdate9;
```

Specify the symbol that references a value to be supplied by the data component. The DYNAMIC statement defines a symbol, Colhd, that references a value that the data component supplies when ODS binds the template and the data component to produce an output object. The values for Colhd, are provided in the FILE statement in the DATA step that appears later in the program. Using dynamic column headings gives you more flexibility than hard-coding the headers in the table template does.

```
dynamic colhd;
```

Control the repetition of values that do not change from one row to the next row. The CLASSLEVELS= attribute suppresses the display of the value in a column that is marked with BLANK_DUPS=ON if the value changes in a previous column that is also marked with BLANK_DUPS=ON. Because BLANK_DUPS= is set in a generic column, set this attribute as well.

```
classlevels=on;
```

Create the column Char_Var. This DEFINE statement and its attributes create the column template Char_Var.

GENERIC= specifies that multiple variables can use the same column template.

BLANK_DUPS= suppresses the display of the value in the column if it does not change from one row to the next (and, because CLASSLEVELS=ON for the table, if no values in preceding columns that are marked with BLANK_DUPS=ON changes).

HEADER= specifies that the header for the column will be the text of the dynamic variable Colhd, whose value will be set by the data component.

The STYLE= attribute specifies that the style element for this column template is CellContents.

The END statement ends the template.

```
define column char_var;
   generic=on;
   blank_dups=on;
   header=colhd;
   style=cellcontents;
end;
```

Create the column Num_Var. This DEFINE statement and its attributes create the column template Num_Var. GENERIC= specifies that multiple variables can use the same column template.

HEADER= specifies that the header for the column will be the text of the dynamic variable Colhd, whose value will be set by the data component.

The STYLE= attribute specifies that the style element for this column template is CellContents.

The END statement ends the template.

```
define column num_var;
   generic=on;
   header=colhd;
```

```
      style=cellcontents;
   end;
```

Create the footer element Table_Footer. The DEFINE statement and its substatement define the table element Table_Footer. The FOOTER argument declares Table_Footer as a footer. The TEXT statement specifies the text of the footer. When ODS binds the data component to the table template (in the DATA step that follows), it will resolve the value of the macro variable SYSDATE9.

```
   define footer table_footer;
      text ''Prepared on '' sysdate9;
   end;
```

End the table template. This END statement ends the table template. The RUN statement executes the PROC TEMPLATE step.

```
   end;
run;
```

Stop the creation of the listing output. The ODS LISTING statement closes the LISTING destination to conserve resources. The LISTING destination is open by default.

```
ods LISTING close;
```

Create the HTML output and specify the style to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file **`newstyle2-body.htm`** in the current directory. The STYLE= option tells ODS to use NewStyle2 as the style when it formats the output.

```
ods html body="newstyle2-body.htm"
         style=newstyle2;
```

Specify the titles for the report. The TITLE statements provide two titles for the output.

```
   title "Leading Grain Producers";
   title2 "in 1996";
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object.

The SET statement reads the data set Grain_Production. The WHERE statement subsets the data set so that the output object contains information only for rice and corn production in 1996.

```
data _null_;
   set grain_production;
   where type  in ("Rice", "Corn") and year=1996;
```

Route the DATA step results to ODS and use the Table1 table template. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, “Output Delivery System and the DATA Step,” on page 39. The TEMPLATE= suboption tells ODS to use the table template named Table1, which was previously created with PROC TEMPLATE.

```
   file print ods=(
        template="table1"
```

Specify the column template to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table template. For example, the first *column-specification* specifies that the first column of the output object contains the values of the variable COUNTRY and that it uses the column template named Char_Var. GENERIC= must be set to ON in both the table template and each column assignment in order for multiple variables to use the same column template. The FORMAT= suboption specifies a format for the column. The DYNAMIC= suboption provides the value of the dynamic variable Colhd for the current column. Notice that for the first column the column header is **`Country`**, and for the second column, which uses the same column template, the column header is **`Year`**.

```
columns=(
   char_var=country(generic=on format=$cntry.
            dynamic=(colhd="Country"))
   char_var=type(generic dynamic=(colhd="Year"))
   num_var=kilotons(generic=on format=comma12.
           dynamic=(colhd="Kilotons"))
   )
);
```

Write the data values to the data component. The _ODS_ option and the PUT statement write the data values for all columns to the data component. The RUN statement executes the DATA step.

```
   put _ods_;
run;
```

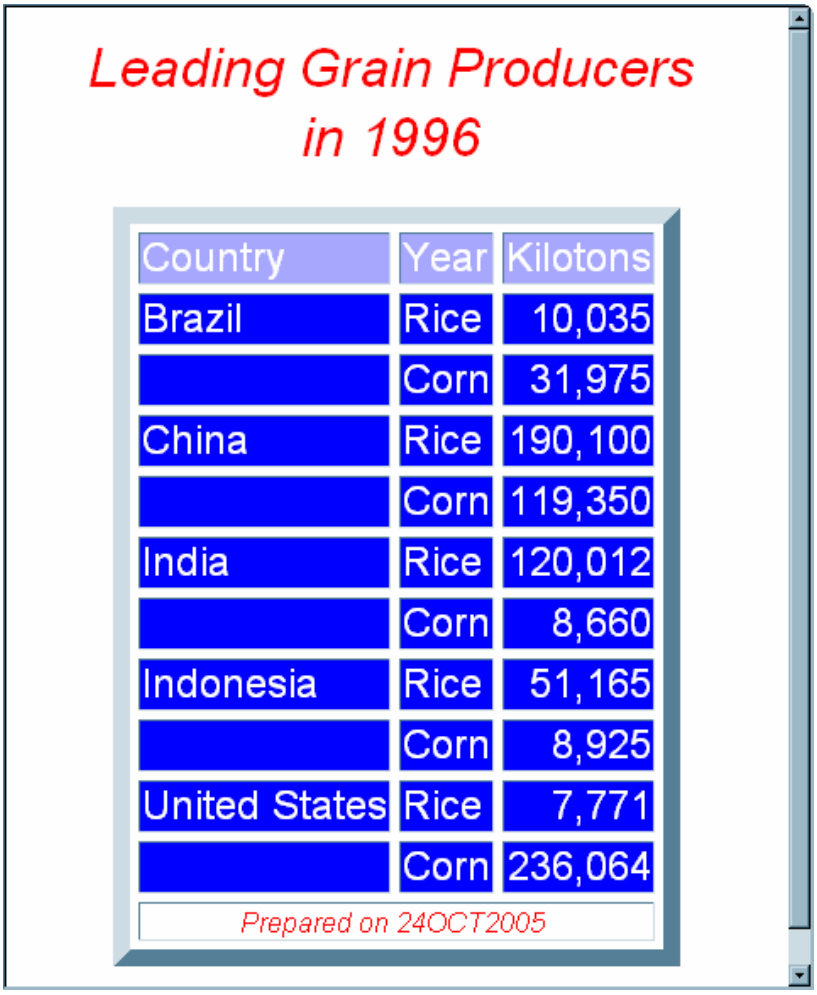
Stop the creation of the HTML output and create the listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. Close the destination so that you can view the output with a browser. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods html close;
ods listing;
```

Original HTML Output

Display 11.11 HTML Output

This HTML output is identical to "HTML Output: Specifying Colors and Fonts with User-Defined Attributes" on page 556, which was produced with a style that used predefined style attributes. You can use the fonts to confirm that SAS titles use the SystemTitle style element, that column headings use the Header style element, that the footer uses the Table-Footer style element, and that the contents of both character and numeric cells use the CellContents style element. Use the width of the table border and the spacing between cells to confirm that the table produced with the Table style element.



Leading Grain Producers in 1996

Country	Year	Kilotons
Brazil	Rice	10,035
	Corn	31,975
China	Rice	190,100
	Corn	119,350
India	Rice	120,012
	Corn	8,660
Indonesia	Rice	51,165
	Corn	8,925
United States	Rice	7,771
	Corn	236,064

Prepared on 24OCT2005

Program 2: Description

In the program Example 1 on page 551, to change the color scheme so that the blues are replaced by pink and red, change each occurrence of "blue" and "very light blue." In this program, because colors are defined as user-defined attributes, make the change only once.

Program 2: Changing User-Defined Attributes

To make the color scheme change, change only this section of code:

```
style colors /
   "light"=white
   "medium"=cxaaaaff
   "dark"=cx0000ff
   "bright"=red;
```

Change the attributes as follows:

```
style colors /
   "light"=white
   "medium"=pink
   "dark"=red
   "bright"=red;
```

Similarly, to change the font in any style element that uses **`cellfont`**, change this section of code:

```
"cellfont"=("arial, helvetica", 4, medium roman)
```

Here is one example of how to change the code:

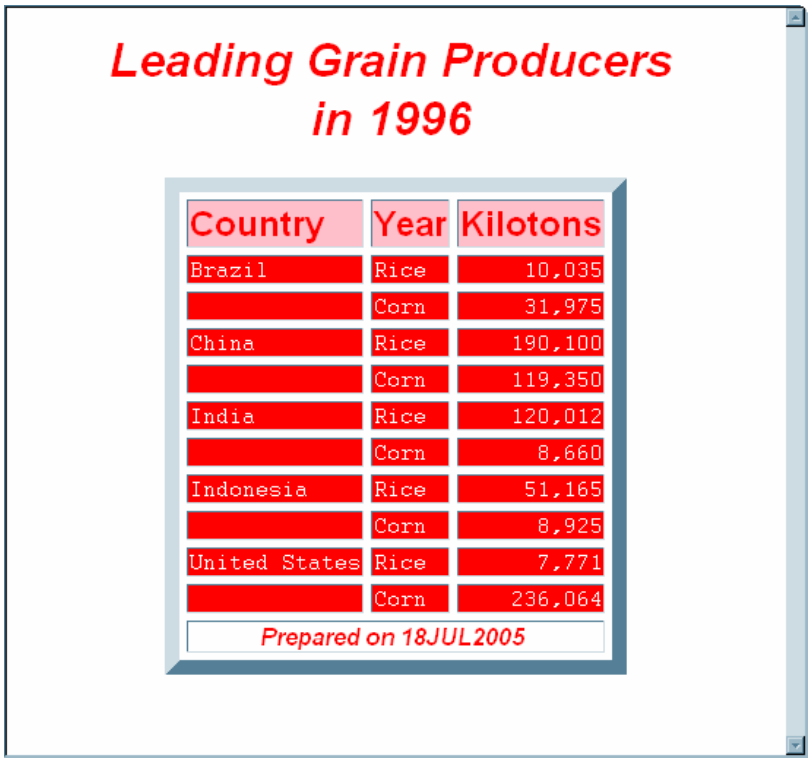
```
"cellfont"=("courier, arial, helvetica", 4, medium roman)
```

This HTML output shows the results of running the same program with these changes.

HTML Output: Changing Colors and Fonts of User-Defined Attributes

Display 11.12 HTML Output with Changed Colors and Fonts

The font in the cells is now Courier. This change occurs in multiple places even though you made only one change to the code for the font.



Leading Grain Producers in 1996

Country	Year	Kilotons
Brazil	Rice	10,035
	Corn	31,975
China	Rice	190,100
	Corn	119,350
India	Rice	120,012
	Corn	8,660
Indonesia	Rice	51,165
	Corn	8,925
United States	Rice	7,771
	Corn	236,064

Prepared on 18JUL2005

Example 3: Using the CLASS Statement

PROC TEMPLATE features:

DEFINE STYLE statement:
- CLASS statement:
- PARENT= statement:
- Style attributes:
  - User-defined attributes
  - BACKGROUNDCOLOR=
  - BORDERWIDTH=
  - CELLPADDING=
  - CELLSPACING=
  - COLOR=
  - FONT=
  - FONTSTYLE=
  - FRAME=
  - POSTHTML=
  - RULES=
  - VISITEDLINKCOLOR=

Other ODS features:

ODS HTML statement:
- STYLE= option

ODS LISTING statement

ODS PATH statement

Data set: See “Creating the Energy Data Set” on page 875.

Formats: See “Creating the DIVFMT. and USETYPE. Formats” on page 872.

Program 1: Description

When you are working with styles, you are more likely to modify a SAS style than to write a completely new style. This example makes changes to the default style for the HTML destination. The new style affects both the contents file and the body file in the HTML output. In the contents file, the modified style makes changes to the following:

- the text of the header and the text that identifies the procedure that produced the output
- the colors for some parts of the text
- the font size of some parts of the text
- the spacing in the list of entries in the table of contents.

In the body file, the modified style makes changes to the following:

- two of the colors in the color list. Style1 of these colors is the foreground color for the table of contents, the byline, and column headings. The other is the foreground of many parts of the body file, including SAS titles and footnotes.
- the font size for titles and footnotes
- the font style for headers
- the presentation of the data in the table by changing attributes such as cellspacing, rules, and border width.

When you modify a style element in a new style that has a like-named style element in the parent style, then you must use the CLASS statement or the STYLE statement with the FROM option specified. This example uses the CLASS statement to produce a shorter, easier to read program.

Program 1: Using the Default Style with PROC PRINT

Specify the search path in order to locate the table template. This statement specifies which locations to search for templates that were created by PROC TEMPLATE, as well as the order in which to search for them. The statement is included to ensure that the example works correctly. However, if you have not changed the path, then you do not need to include this statement because it specifies the default path.

```
ods path sasuser.templat(update) sashelp.tmplmst(read);
```

Stop the creation of the listing output. The ODS LISTING statement closes the LISTING destination to conserve resources. The LISTING destination is open by default.

```
ods listing close;
```

Create the HTML output and specify the name of the HTML file. Specify the style to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file. FRAME= and CONTENTS= create a frame that includes a table of contents that links to the contents of the body file. The body file also appears in the frame.

The STYLE= option tells ODS to use Styles.Default as the style when it formats the output. Strictly speaking, this option is unnecessary because it specifies the default style, but it is included for clarity.

```
ods html body="sasdefaultstyle-body.htm"
         contents="sasdefaultstyle-content.htm"
         frame="sasdefaultstyle-frame.htm"
         style=styles.default;
```

Specify the titles and footnote for the report. The TITLE and FOOTNOTE statements provide two titles and a footnote for the output.

```
title "Energy Expenditures for Each Region";
title2 "(millions of dollars)";
```

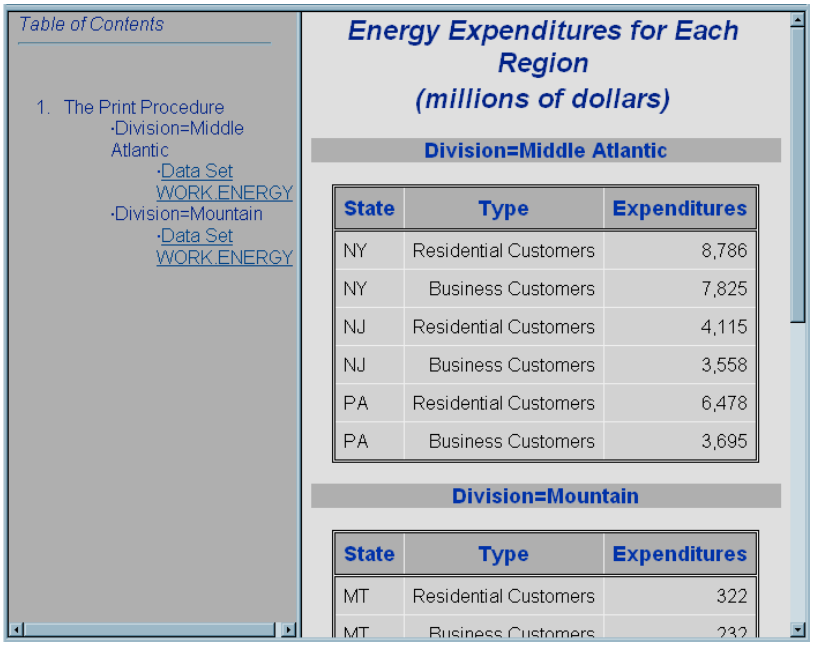
Print the report. PROC PRINT creates a report that includes three variables. ODS writes the report to the BODY file.

```
proc print data=energy noobs;
   var state type expenditures;
   format division divfmt. type usetype. expenditures comma12.;
   by division;
   where division=2 or division=3;
run;
```

Stop the creation of the HTML output and initiate the creation of listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. Close the destination so that you can view the output with a browser. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods html close;
ods listing;
```

Display 11.13 HTML Output from PROC PRINT Using the Default Style



Program 2: Modifying the Default Style with the CLASS Statement

Create the style CustomDefault. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called CustomDefault.

```
proc template;
  define style customdefault;
```

Specify the parent style from which the CustomDefault style inherits its attributes. The PARENT= attribute specifies Styles.Default as the style from which the current style inherits. All the style elements, attributes, and statements that are specified in the parent's style template are used in the child style template unless the child style template overrides them.

```
    parent=styles.default;
```

Change the attributes of the style element Color_List. This STYLE statement adds to the child style the style element Color_List, which also exists in the parent style. The CLASS stament adds all of the style attributes that are in the original instance of the Color_List style element to the new instance of Color_List, except for those that are overriden by the new instance of Color_List. By using the CLASS statement, you do not need to specify the FROM option.

Note: If you did not use the CLASS statement or the FROM option, then the attributes from the original instance of Color_List would not be added to the new instance of Color_List. The Color_List style element in CustomDefault would conatain only the style statements that it specifically specifies. △

All style elements that use the user-defined attributes that Color_List defines (**fgB2**, **fgB1**, etc.) use the style attributes that are specified in Custom.Default, not the ones that are specified in Styles.Default. Therefore, if you change a color here, then you change every occurrence of the color in the HTML output. This CLASS statement changes the values of five of the user-defined style attributes.

```
   class color_list/
/* changed from cxD3D3D3*/
      "bgA3" = #778899
/* changed from cx0033AA */
      "fgA2" = #68228b
/* changed from cxB0B0B0 */
      "bgA2" = #fff8dc
/* changed from cx000000 */
      "fgA1" = #fff8dc
/* changed from cxE0E0E0 */
      "bgA" = #c6e2ff;
```

Change the attributes of the style element TitlesandFooters. This CLASS statement adds to the child style the style element TitlesandFooters, which also exists in the parent style. By using the CLASS statement, you do not need to specify the FROM option. The style attributes in the new instance of TitlesandFooters override the style attributes from the parent style element, which is TitlesandFooters from Styles.Default. The new instance of TitlesandFooters will pass its attributes to any style element that inherits from it. This style element uses **systitlefg** and **systitlebg** for colors, but it changes the font size from the relative size of 4 that is specified in TitleFont2 to a relative size of 3. As a result, the titles and footnotes in Display 11.14 on page 570 are smaller than the ones in Display 11.13 on page 566.

```
class titlesandfooters/
   color=colors("systitlefg")
   backgroundcolor=colors("systitlebg")
   font=fonts("titlefont2") fontsize=3;
```

Change the attributes of the style element Byline. Specify that the style element Byline inherits its attributes from the TitlesandFooters style element. This STYLE statement adds to the child style the style element Byline, which also exists in the parent style. This style element inherits all attributes from TitlesandFooters as it is specified in the previous STYLE statement. Therefore, the initial template for the byline includes the font colors and background colors that are used for system titles, and a smaller version of TitleFont2. However, the COLOR= attribute overrides the font color with the headers" font color. In the default style, the background color for the byline differs from the background color for the document, so it appears as a gray stripe in Display 11.13 on page 566. In this customized style, the stripe disappears because the background color for the byline and the document are the same.

```
style byline from titlesandfooters /
   color=colors("headerfg");
```

Change one attribute in the style Header. This STYLE statement adds the italic font style to the attributes that Header inherits from the Header style element that is defined in the parent style. You could have also specified the STYLE statement with the FROM option specified. Because this change occurs after the initial merge of the two styles, the change will effect HeaderFixed and the other style elements that inherit from Header in the parent style.

```
class header  /
   fontstyle=italic;
```

Customize the text used in parts of the output. This CLASS statement alters the text that is used in parts of the HTML output. In the contents file, the default style uses “The” as the value of **`prefix1`** and “Procedure” as the value of **`suffix1`**. Thus, in HTML output that uses the default style, the output from PROC PRINT is identified by “1. The PRINT Procedure” (see Display 11.13 on page 566). In the customized style, the text that identifies the output reads “1. PROC PRINT”. The heading that appears at the top of the contents file has been changed from “Table of Contents” to “Contents”, and the heading at the top of the table of pages has been changed from “Table of Pages” to “Pages”. The banners have been changed to use mixed case. (Note that neither these banners nor the table of pages is visible in the HTML output from this example, but the attributes are included so that you can use the style in a variety of circumstances.)

```
class text /
   "prefix1" = "PROC "
   "suffix1" = ":"
   "Content Title" = "Contents"
   "Pages Title" = "Pages"
   "Note Banner" = "Note:"
   "Warn Banner" = "Warning:"
   "Error Banner" = "Error:"
   "Fatal Banner" = "Fatal:"
   ;
```

Customize the presentation of the HTML table that contains the output from PROC PRINT. This CLASS statement changes the presentation of the HTML table that contains the output from PROC PRINT. The background color, the kind of box that surrounds the table, and the cell padding remain the same as in Styles.Default, but all the other attributes are changed. RULES=COLS draws rules only between the columns of the table. CELLSPACING=0 removes the spacing between the cells of the table so that the data appear on a continuous background. BORDERWIDTH= increases the width of the table's border. The changes dramatically alter the appearance of the HTML output.

```
   class table /
      rules=cols
      cellspacing=0
      borderwidth=5;
```

Change the color of links and the foreground. This CLASS statement changes the value of the VISITEDLINKCOLOR= attribute in the style element Contents so that the links in the table of contents appear in the same color as the rest of the table of contents. It also changes the foreground color so that the title of the table of contents appears in the same color as system titles.

```
   class contents /
      visitedlinkcolor=colors('systitlefg")
      color=colors('systitlefg");
```

Add more space between the items in the table of contents. This CLASS statement adds the POSTHTML= attribute so that the items in the table of contents are displayed with extra space between them.

```
   class contentitem /
      posthtml="<p>";
```

Stop the creation of the customized style. The END statement ends the style. The RUN statement executes the PROC TEMPLATE step.

```
   end;
run;
```

Create the HTML output and specify the specify the style to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file. FRAME= and CONTENTS= create a frame that includes a table of contents that links to the contents of the body file. The body file also appears in the frame.

The STYLE= option tells ODS to use CustomDefault as the style when it formats the output.

```
ods html body="customdefaultstyle-body.htm"
         contents="customdefaultstyle-content.htm"
         frame="customdefaultstyle-frame.htm"
         style=customdefault;
```

Specify the titles and footnote for the report. The TITLE and FOOTNOTE statements provide two titles and a footnote for the output.

```
title "Energy Expenditures for Each Region";
title2 "(millions of dollars)";
```

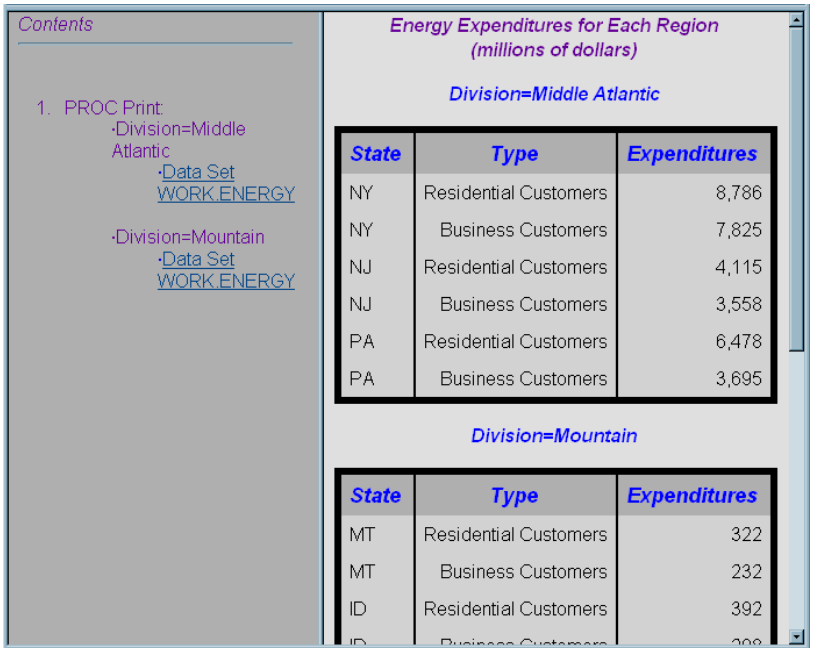
Print the customized report. PROC PRINT creates a report that includes three variables. ODS writes the report to the body file. This PROC PRINT step is the same one that was used with the default style earlier.

```
proc print data=energy noobs;
   var state type expenditures;
   format

 division divfmt. type usetype. expenditures comma12.;
   by division;
   where division=2 or division=3;
run;
```

Stop the creation of the HTML output and initiate the creation of listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. Close the destination so that you can view the output with a browser. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods html close;
ods listing;
```

Display 11.14 HTML Output from PROC PRINT with the Customized Style



Example 4: Defining a Table and Graph Style

PROC TEMPLATE features:

DEFINE STYLE statement:

PARENT= attribute

STYLE statement

Style attributes:

User-defined attributes

BACKGROUNDCOLOR=
BORDERCOLORDARK=
BORDERCOLORLIGHT=
BORDERWIDTH=
CELLPADDING=
CELLSPACING=
DROPSHADOW=
ENDCOLOR=
FONT=
COLOR=
FRAME=
GRADIENTDIRECTION=
IMAGE=
TEXTALIGN=
WIDTH=
RULES=
TRANSPARENCY=
VERTICALALIGN=

Style elements:

GraphAxisLines
GraphBackground
GraphBorderLines
GraphCharts
GraphLabelText
GraphWalls

Program Description

When you are working with styles, you are more likely to modify a SAS style than to write a completely new style. This example shows you how the SAS defined graph style, Science, was created.

Note: Remember that when a STYLE statement creates a style element in the new style, only style elements that explicitly inherit from that style element in the new style inherit the change. When a STYLE statement creates a style element in the new style, all style elements that inherit from that element inherit the definition that is in the new style, so the change appears in all children of the element. △

Program

Create the style Science. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style in the STYLES catalog called Science.

```
proc template;
     define style Styles.Science;
```

Specify the parent style from which the Science style inherits its attributes. The PARENT= attribute specifies Styles.Default as the style that the current style inherits from. All the style elements that are specified in the parent's style are used in the current style unless the current style overrides them.

```
       parent = styles.default;
```

Change the style attributes of the Fonts style element in the parent style by replacing Fonts in the child style Science. The STYLE statement adds to the child style the style element Fonts, which also exist in the parent style. All style elements that use the user-defined attributes that Fonts define use the attributes that are specified in the STYLE statement, not the ones that are specified in the Styles.Default style. Because no FROM option is specified, the instance of Fonts in the Science style completely replaces the instance from the Styles.Default style. No style element inheritance occurs.

```
style fonts /
   "TitleFont2" = ("Verdana, Verdana, Helvetica, sans-serif",14pt,Bold)
   "TitleFont" = ("Verdana, Verdana, Helvetica, sans-serif",18pt,Bold)
   "StrongFont" = ("Verdana, Verdana, Helvetica, sans-serif",14pt,Bold)
   "EmphasisFont" = ("Verdana, Verdana, Helvetica, sans-serif",10pt,
   Italic)
   "FixedEmphasisFont" = (""Courier New", Courier, monospace",10pt,
   Italic)
   "FixedStrongFont" = (""Courier New", Courier, monospace",10pt,Bold)
   "FixedHeadingFont" = (""Courier New", Courier, monospace",10pt)
   "BatchFixedFont" = (""Courier New", Courier, monospace",10pt)
   "FixedFont" = (""Courier New", Courier, monospace",10pt)
   "headingEmphasisFont" = ("Verdana, Verdana, Helvetica, sans-serif",14
   pt,Bold Italic)
   "headingFont" = ("Verdana, Verdana, Helvetica, sans-serif",14pt,Bold)

   "docFont" = ("Verdana, Verdana, Helvetica, sans-serif",8pt,Bold);
```

Change the attributes for graph style specific fonts. The STYLE statement adds to the child styles the style element GraphFonts, which also exists in the parent style. All the style elements that use the user-defined attributes that GraphFonts defines use the attributes specified in the STYLE statement, not those specified in the STYLES.DEFAULT style. Because the FROM option is specified, GraphFonts in the Science style will inherit all of the style attributes from GraphFonts in Styles.Default, except those that specifically specified in Science.

Note: Instead of the one that is used in this program, you could have used the following STYLE statement :

```
style graphfaonts from graphfonts;
```

△

```
style GraphFonts from _self_/
         "GraphValueFont" = ("Verdana",10pt)
         "GraphLabelFont" = ("Verdana",14pt,Bold);
```

Change the style attributes of the Colors style element in the parent style by replacing Colors in the style Science. The STYLE statement adds to the child styles the style element Colors, which also exists in the parent style. All style elements that use the user-defined attributes that Colors defines use the attributes that are specified in the STYLE statement, not the ones that are specified in the Styles.Default style. Because no FROM option is specified, the instance of Colors in the Science style completely replaces the instance from the Styles.Default style. No style element inheritance occurs.

```
style colors /
    "headerfgemph" = cx31035E
    "headerbgemph" = cxFFFFFF
    "headerfgstrong" = cx31035E
    "headerbgstrong" = cxFFFFFF
```

```
        "headerfg" = cx31035E
        "headerbg" = cxFFFFFF
        "datafgemph" = cx31035E
        "databgemph" = cxDFECE1
        "datafgstrong" = cx31035E
        "databgstrong" = cxDFECE1
        "datafg" = cx31035E
        "databg" = cxDFECE1
        "batchfg" = cx31035E
        "batchbg" = cxDFECE1
        "tablebg" = cx31035E
        "tableborderdark" = cx909090
        "tableborderlight" = cxFFFFFF
        "tableborder" = cxFFFFFF
        "notefg" = cx31035E
        "notebg" = cxDFECE1
        "bylinefg" = cx31035E
        "bylinebg" = cxDFECE1
        "captionfg" = cx31035E
        "captionbg" = cxDFECE1
        "proctitlefg" = cx31035E
        "proctitlebg" = cxDFECE1
        "titlefg" = cx31035E
        "titlebg" = cxDFECE1
        "systitlefg" = cx31035E
        "systitlebg" = cxDFECE1
        "Conentryfg" = cx31035E
        "Confolderfg" = cx31035E
        "Contitlefg" = cx31035E
        "link2" = cx800080
        "link1" = cx0000FF
        "contentfg" = cx31035E
        "contentbg" = cxDFECE1
        "docfg" = cx31035E
        "docbg" = cxDFECE1;
```

Change the style attributes for the GraphColors style element. The STYLE statement adds to the child styles the style element GraphColors, which also exists in the parent style. All of the style elements that use the user-defined attributes that GraphColors define use the attributes that are specified in the Science style, not the attributes that are specified in the Styles.Default style. Because no FROM option is specified, the instance of GraphColors in the Science style completely replaces the instance from the Styles.Default style. No style element inheritance occurs.

```
 style GraphColors /
        "gconramp3cend" = cxDD6060
        "gconramp3cneutral" = cxFFFFFF
        "gconramp3cstart" = cx6497EB
        "gramp3cend" = cxBED8D3
        "gramp3cneutral" = cxFFFFFF
        "gramp3cstart" = cxAAB6DF
        "gconramp2cend" = cx6497EB
        "gconramp2cstart" = cxFFFFFF
        "gramp2cend" = cx548287
        "gramp2cstart" = cxFFFFFF
```

```
    "gtext" = CX31035E
    "glabel" = CX31035E
    "gborderlines" = CX31035E
    "goutlines" = CX31035E
    "ggrid" = CX31035E
    "gaxis" = CX31035E
    "gshadow" = CX707671
    "glegend" = CXFFFFFF
    "gfloor" = CXDFECE1
    "gwalls" = CXFFFFFF
    "gcdata12" = cxFF667F
    "gcdata11" = cx5050CC
    "gcdata10" = cxE100BF
    "gcdata9" = cx007F00
    "gcdata8" = cxB99600
    "gcdata7" = cx7F7F7F
    "gcdata6" = cx984EA3
    "gcdata5" = cx4DAF4A
    "gcdata4" = cxA65628
    "gcdata3" = cxFF7F00
    "gcdata2" = cx377DB8
    "gcdata1" = cxE31A1C
    "gdata12" = CX4A5573
    "gdata11" = CXCFB1E2
    "gdata10" = CX8E829D
    "gdata9" = CX2952B1
    "gdata8" = CXAAB6DF
    "gdata7" = CX6771C2
    "gdata6" = CXBED8D3
    "gdata5" = CX8B65A3
    "gdata4" = CXBCD3AB
    "gdata3" = CX548287
    "gdata2" = CX7DC1C9
    "gdata1" = CX9580D5;
```

Specify attributes for the table. This STYLE statement is applied to tables. Although these specific attributes are set with this STYLE statement, all other table attributes are inherited from the style elements that are defined in the parent styles.

```
style Table from Output /
    cellpadding = 5
    cellspacing = 2
    bordercolordark = colors("tableborderdark")
    bordercolorlight = colors("tableborderlight")
    borderwidth = 2;
```

Specify attributes for the GraphLabelText element. This STYLE statement is applied to the graph's label text. A DROPSHADOW attribute is applied.

```
style GraphLabelText from GraphLabelText
    "Label attributes" /
    dropshadow = on;
```

Replace the background for the Graph. This STYLE statement is applied to the graph's background. **DOCBG** is specified as the background colors, with SCIENCE.GIF justified to the left and bottom as the background image.

```
 style GraphBackground
    "Graph backgroundcolor attributes" /
    backgroundcolor = colors("docbg")
    image = "!sasroot\common\textures\Science.gif"
    textalign = L
    verticalalign = B;
```

Specify attributes for the GraphAxisLines element. This STYLE statement is applied to the graph's axis line. The WIDTH is **2**.

```
style GraphAxisLines from GraphAxisLines
    "Axis line attributes" /
    width = 2;
```

Specify attributes for the GraphBorderLines element. This STYLE statement is applied to the border lines in the graph. The width is **2** and the foreground color defined in Gaxis, which is CX31035E, is used.

```
style GraphBorderLines from GraphBorderLines
    "Border attributes" /
    width = 2
    color=colors("gaxis");
```

Specify attributes for the GraphCharts element. This STYLE statement is applied to the graph's chart. The data elements of the graph have a TRANSPARENCY of 25 percent.

```
style GraphCharts from GraphCharts
    "Chart Attributes" /
    transparency = 0.25;
```

Specify attributes for the GraphWalls element. This STYLE statement is applied to the walls inside of the graph's axes. The GRADIENTDIRECTION is set to **Xaxis**, meaning the gradient is going left to right. The ENDCOLOR, defined in Gwalls, which is CXFFFFFF, is the final color used with the gradient. The data elements of the graph have a TRANSPARENCY of 100 percent. Because a STARTCOLOR is not specified, the beginning of the gradient is completely transparent.

```
style GraphWalls from GraphWalls
    "Wall Attributes" /
    gradientdirection = "Xaxis"
    endcolor = colors("gwalls")
    transparency = 1.0;
```

Add the style to the specified catalog. The END statement ends the style. The RUN statement executes the PROC TEMPLATE step.

```
   end;
run;
```

Example 5: Defining Multiple Style Elements in One STYLE Statement

PROC TEMPLATE features:

DEFINE STYLE statement:
- STYLE statement:
  - FROM option
  - Style attributes:
    - BACKGROUNDCOLOR=
    - BORDERWIDTH=
    - CELLSPACING=
    - FONTFAMILY=
    - FONTSIZE=
    - FONTSTYLE=
    - FONTWEIGHT=
    - COLOR=

DEFINE TABLE statement:
- CLASSLEVELS= table attribute
- DYNAMIC statement
- MVAR statement

DEFINE COLUMN statement:
- BLANK_DUPS=
- GENERIC=
- HEADER=
- STYLE=

DEFINE FOOTER statement:
- TEXT statement

Other ODS features:

- ODS HTML statement
- ODS LISTING statement
- FILE statement with ODS= option
- PUT statement with _ODS_ argument

Data set: See "Creating the Grain_Production Data Set" on page 878.

Format: See "Creating the $CNTRY Format" on page 869.

Table template: See "Creating the Table1 Table Definition" on page 882.

Program Description

This example creates a style that defines multiple style elements concurrently. When style element names are specified multiple times, all of the attributes from all instances of that name are collected to create the final set of style attributes. Defining multiple style elements in one STYLE statement makes it possible to create shorter, easier to read programs and to make changes to style attributes in a single STYLE statement rather than in many STYLE statements.

For example, if you wanted to add the style element BorderColor=black to the style elements CellContents, Header, and SystemTitle in the program below, you could add it once, to the first STYLE statement, instead of adding it three times, to each individual STYLE statement.

Program

Create a new style NewStyle. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called NewStyle.

```
proc template;
   define style newstyle;
```

Create the CellContents, Header, and SystemTitle style elements.

This STYLE statement defines three style elements: CellContents, Header, and SystemTitle. They are all composed of the style attributes that appear on the STYLE statement.

The FONTFAMILY= attribute tells the browser to use the Arial font if it is available, and to look for the Helvetica font if Arial is not available. These three syle elements use a color scheme of a white foreground on a blue background, and the font for all three is medium roman with a size of five.

```
      style cellcontents, header, systemtitle /
         fontfamily="arial, helvetica"
         fontweight=medium
         backgroundcolor=blue
         fontstyle=roman
         fontsize=5
         color=white;
```

Modify the Header style element. The STYLE statement with the FROM option specified creates the new instance of Header from the previous instance of Header, but changes the background color from white to very light blue. By default, ODS uses Header to produce both spanning headers and column headings.

```
      class header /
         backgroundcolor=very light blue;
```

Modify the SystemTitle style element. By default, ODS uses SystemTitle to produce SAS titles.

```
      class systemtitle /
         backgroundcolor=white
         color=red
         fontstyle=italic
         fontsize=6;
```

Create the style element Footer. This STYLE statement creates the style element Footer. This style element inherits all the attributes of SystemTitle. However, the font size that it inherits is overwritten by the FONTSIZE= attribute in its template.

```
      style footer from systemtitle /
         fontsize=3;
```

Create the style element Table. This STYLE statement creates the style element Table. By default, ODS uses this style element to display tables.

```
      class table /
         cellspacing=5
         borderwidth=10;
      end;
run;
```

Stop the creation of the listing output. The ODS LISTING statement closes the LISTING destination in order to conserve resources. The LISTING destination is open by default.

```
ods listing close;
```

Create HTML output and specify the location for storing the HTML output. Specify the style to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file NewStyle-Body in the current directory. The STYLE= option tells ODS to use NewStyle as the style when it formats the output.

```
ods html body="newstyle-body.htm"
         style=newstyle;
```

Specify the titles for the report. The TITLE statements provide two titles for the output.

```
   title "Leading Grain Producers";
   title2 "in 1996";
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object.

The SET statement reads the data set Grain_Production. The WHERE statement subsets the data set so that the output object contains information only for rice and corn production in 1996.

```
data _null_;
   set grain_production;
   where type  in ("Rice", "Corn") and year=1996;
```

Route the DATA step results to ODS and use the Table1 table template. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, "Output Delivery System and the DATA Step," on page 39.) The TEMPLATE= suboption tells ODS to use the table template named Table1, which was previously created with PROC TEMPLATE.

```
   file print ods=(
        template="table1"
```

Specify the column template to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table template. For example, the first *column-specification* specifies that the first column of the output object contains the values of the variable COUNTRY and that it uses the column template named Char_Var. GENERIC= must be set to ON in both the table template and each column assignment in order for multiple variables to use the same column template. The FORMAT= suboption specifies a format for the column. The DYNAMIC= suboption provides the value of the dynamic variable Colhd for the current column. Notice that for the first column the column header is Country, and for the second column, which uses the same column template, the column header is Year.

```
        columns=(
           char_var=country(generic=on format=$cntry.
                    dynamic=(colhd="Country"))
           char_var=type(generic dynamic=(colhd="Year"))
           num_var=kilotons(generic=on format=comma12.
                   dynamic=(colhd="Kilotons"))
           )
        );
```

Write the data values to the data component. The _ODS_ option and the PUT statement write the data values for all columns to the data component. The RUN statement executes the DATA step.

```
   put _ods_;
run;
```

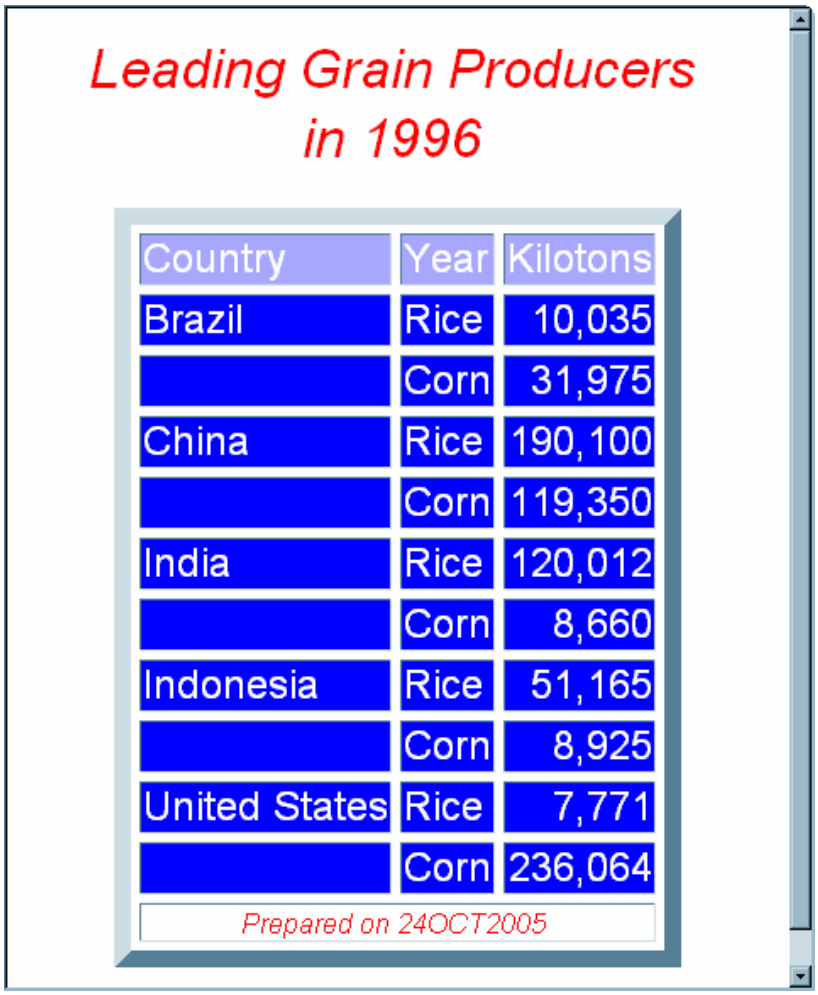
Stop the creation of the HTML output and create the listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. Close the destination so that you can view the output with a browser. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods html close;
ods listing;
```

HTML Output: Specifying Colors and Fonts

Display 11.15 HTML Output

You can use the fonts to confirm that SAS titles use the SystemTitle style element, that column headings use the Header style element, that the footer uses the Table-Footer style element, and that the contents of both character and numeric cells use the CellContents style element. Use the width of the table border and the spacing between cells to confirm that the table itself is produced with the Table style element.



Leading Grain Producers in 1996

Country	Year	Kilotons
Brazil	Rice	10,035
	Corn	31,975
China	Rice	190,100
	Corn	119,350
India	Rice	120,012
	Corn	8,660
Indonesia	Rice	51,165
	Corn	8,925
United States	Rice	7,771
	Corn	236,064

Prepared on 24OCT2005

Example 6: Importing a CSS file

PROC TEMPLATE features:

DEFINE STYLE statement:

CLASS statement

IMPORT statement:

media-type

PARENT= statement

Other ODS features:

ODS HTML statement

ODS PDF statement

ODS _ALL_ CLOSE statement

Program Description

The following program imports the external CSS file StyleSheet.css and converts the CSS code into style elements and style attributes. These style elements and attributes then become part of the style.

Your CSS file can contain media blocks that correspond to the type of media that your output will be rendered on. The IMPORT statement allows you to specify one or more media blocks to be imported along with the rest of the CSS code. In this example, the Print media block is included in the style that is applied to the PDF output.

Program

Example CSS file. The following code is an example of the external CSS file StyleSheet.css. There are two media type blocks specified in this program, Print and Screen.

```
.body {
   background-color: white;
    color: black;
    font-family: times, serif;
}
.header, .rowheader, .footer, .rowfooter, .data {
    border: 1px black solid;
    color: black;
    padding: 5px;
    font-family: times, serif;
}
.header, .rowheader, .footer, .rowfooter {
    background-color: #a0a0a0;
}
.table {
    background-color: #dddddd;
    border-spacing: 0;
    border: 1px black solid;
}
.proctitle {
    font-family: helvetica, sans-serif;
```

```
    font-size: x-large;
    font-weight: normal;
}

@media screen {

   .header, .rowheader, .footer, .rowfooter,{
      color: white;
      background-color: green;}
   .table {
      background-color: yellow;
      border-spacing: 0;
      font-size: small
      border: 1px black solid;

}
}@media print {

  .header, .rowheader, .footer, .rowfooter,{
      color: white;
      background-color: Blue;
      padding: 5px;
}
  .data {
      font-size: small;
}
}


options nodate pageno=1 linesize=80 pagesize=40 obs=10;
```

Define a style that imports a CSS file and defines style elements as well. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called MyCssStyle. The IMPORT statement imports the CSS file StyleSheet.css, and converts the CSS code into ODS style elements and style attributes. Because no *media-type* option is specified, the Screen media block is imported along with the CSS code that is not in any media blocks. The Print media block is not imported. The CLASS statement specifies a red font color in the Data style element.

Note: Specifying:

```
class data / color=red;
```

is the same as specifying:

```
style data from data / color=red;
```

△

```
proc template;
define style styles.mycssstyle;
    import "StyleSheet.css";
    class data /
        color = red;
end;
```

Define a style that imports a CSS file that includes a specific media type templates. The DEFINE STYLE statement creates a new style called MyCssStylePrinter. The IMPORT statement imports the CSS file StyleSheet.css, and converts the CSS code into ODS style elements and style attributes. The Print option specifies that the Print media block be imported along with the CSS code that is not in any media blocks. The code in the Screen media block is not imported.

```
define style styles.mycssstyleprinter;
        parent=styles.mycssstyle;
        import "StyleSheet.css" print;
end;
run;
```

Create HTML and PDF output and view the contents of the SAS data set. The ODS HTML and ODS PDF statements specify the destination to write to, the file name of the output, and the style to use. The CONTENTS procedure shows the contents of the SAS data set SasHelp.Class.

```
ods html file="css.html" style=styles.mycssstyle;
ods pdf file="css.pdf" style=styles.mycssstyleprinter;

proc contents data=sashelp.class;
run;
```

Close the open destinations. The ODS _ALL_ CLOSE statement closes all open destinations and the files that are associated with them. If you do not close the destinations, then you will not be able to view the files.

```
ods _all_ close;
```

Output

Output 11.1 MyCssStyle Style

```
proc template;
   define style Styles.Mycssstyle / store = SASUSER.TEMPLAT;
      class body /
         fontfamily = "times, serif"
         color = #000000
         backgroundcolor = #FFFFFF;
      class header /
         backgroundcolor = #008000
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #FFFFFF
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class rowheader /
         backgroundcolor = #008000
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #FFFFFF
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class footer /
         backgroundcolor = #008000
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #FFFFFF
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
```

```
   class rowfooter /
         backgroundcolor = #A0A0A0
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #000000
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class data /
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = red
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class table /
         fontsize = 3
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px
         borderspacing = 0
         backgroundcolor = #FFFF00;
      class proctitle /
         fontweight = medium
         fontsize = 6
         fontfamily = "helvetica, sans-serif";
   end;
run;
```

Display 11.16 MyCssStyle Style Applied to HTML Output

The yellow and green background colors, the white font color, the font size and border information all come from the Screen media block. The red font color comes from the CLASS statement. All other style information comes from the code outside of the media blocks. No information from the Print media block is used.

The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Thursday, May 19, 2005 03:25:02 PM	Observation Length	40
Last Modified	Thursday, May 19, 2005 03:25:02 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
File Name	C:\SASv9\sasgen\dev\nmva-v920\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0201B0
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

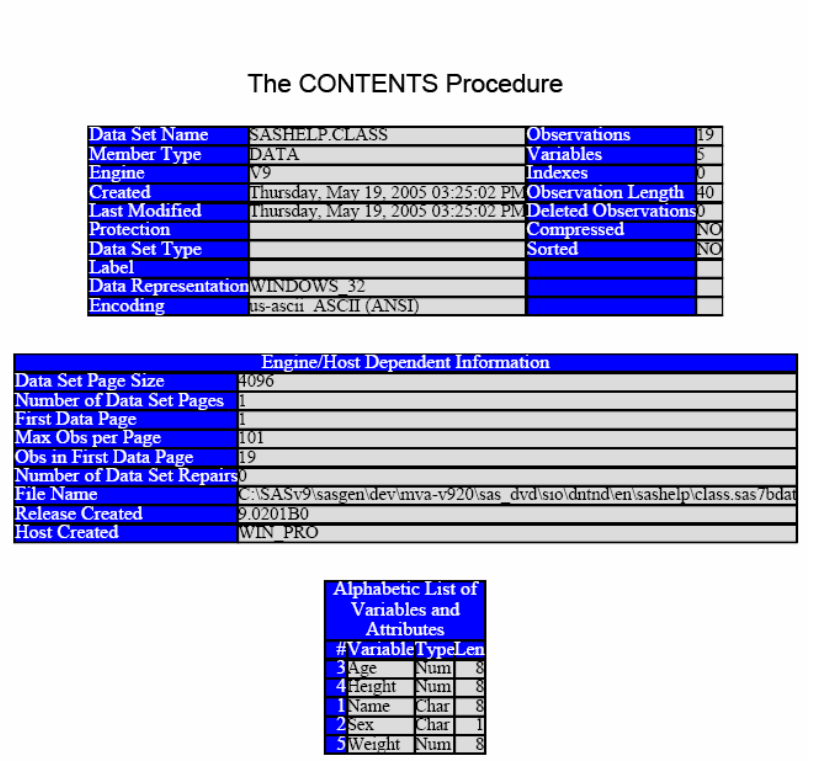
Output 11.2 MyCssStylePrinter Style

```
proc template;
   define style Styles.Mycssstyleprinter / store = SASUSER.TEMPLAT;
      parent = styles.mycssstyle;
      class body /
         fontfamily = "times, serif"
         color = #000000
         backgroundcolor = #FFFFFF;
      class header /
         backgroundcolor = #0000FF
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #FFFFFF
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class rowheader /
         backgroundcolor = #0000FF
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #FFFFFF
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class footer /
         backgroundcolor = #0000FF
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #FFFFFF
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
```

```
      class rowfooter /
         backgroundcolor = #A0A0A0
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #000000
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class data /
         fontsize = 3
         fontfamily = "times, serif"
         paddingleft = 5px
         paddingbottom = 5px
         paddingright = 5px
         paddingtop = 5px
         color = #000000
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px;
      class table /
         borderleftstyle = solid
         borderleftcolor = #000000
         borderleftwidth = 1px
         borderbottomstyle = solid
         borderbottomcolor = #000000
         borderbottomwidth = 1px
         borderrightstyle = solid
         borderrightcolor = #000000
         borderrightwidth = 1px
         bordertopstyle = solid
         bordertopcolor = #000000
         bordertopwidth = 1px
         borderspacing = 0
         backgroundcolor = #DDDDDD;
      class proctitle /
         fontweight = medium
         fontsize = 6
         fontfamily = "helvetica, sans-serif";
   end;
run;
```

Display 11.17 MyCssStylePrinter Style Applied to PDF Output

The white font, small font size, cell padding, and the blue background color all come from the Print media block. All other style information comes from the code outside of the media blocks. No information from the Screen media block is used.



The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Thursday, May 19, 2005 03:25:02 PM	Observation Length	40
Last Modified	Thursday, May 19, 2005 03:25:02 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
File Name	C:\SASv9\sasgen\dev\mva-v920\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0201B0
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Example 7: Table Header and Footer Border Formatting

PROC TEMPLATE features:

Border control style attributes:

BORDERBOTTOMCOLOR=
BORDERBOTTOMSTYLE=
BORDERBOTTOMWIDTH=
BORDERTOPCOLOR=
BORDERTOPSTYLE=
BORDERTOPWIDTH=

DEFINE statement
DEFINE STYLE statement
EDIT statement
FOOTER statement
HEADER statement
PARENT= statement
PREFORMATTED= header attribute

STYLE statement

WIDTH= header attribute

Other ODS features:

ODS RTF

ODS SELECT

Data set: See “Creating the Nlits Data Set” on page 889.

Program Description

You can use the TableHeaderContainer and TableFooterContainer style elements along with the border control style attributes to change the borders of the regions surrounding the table header and footer.

Note: The TableHeaderContainer and TableFooterContainer style elements are only valid in the RTF destination. △

Program

Set the SAS system options and specify titles. The OPTIONS statement sets the SAS system options and the TITLE statements specify titles for the output.

```
options nodate nonumber;

ods listing close;

title  "TableHeaderContainer, TableFooterContainer, and Border Control Style
     Attributes";
title2 "Allows Control of Borders Between the Header, Body, and Footer of a
     Table";
```

Create the new style HeadersFootersBorders. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style HeadersFootersBorders. The PARENT= statement specifies that the new style inherits all of its style elements and style attributes from the Styles.RTF style.

```
proc template;
   define style  HeadersFootersBorders;
   parent=styles.rtf;
```

Modify the TableHeaderContainer style element. The STYLE statement with the FROM option specified creates the style element TableHeaderContainer which inherits all of its style elements and style attributes from the instance of TableHeaderContainer in the Styles.RTF style. The BORDERBOTTOMWIDTH=, BORDERBOTTOMCOLOR=, and BORDERBOTTOMSTYLE= style attributes specify the width, color, and line style of the bottom border of the table header.

```
   style TableHeaderContainer from TableHeaderContainer /
      borderbottomwidth=12
      borderbottomcolor=blue
      borderbottomstyle=dotted;
```

Modify the TableFooterContainer style element. The STYLE statement with the FROM option specified creates the style element TableFooterContainer which inherits all of its style elements and style attributes from the instance of TableFooterContainer in the Styles.RTF style. The BORDERTOPWIDTH=, BORDERTOPCOLOR=, and BORDERTOPSTYLE= style attributes specify the width, color, and line style of the top border of the table footer.

```
    style TableFooterContainer from TableFooterContainer /
       bordertopwidth=6
       bordertopcolor=red
       bordertopstyle=double;
```

Modify the Table style element. The STYLE statement with the FROM option specified creates the style element Table which inherits all of its style elements and style attributes from the instance of Table in the Styles.RTF style. The CELLSPACING=, RULES=, and FRAME= attributes modify the cellspacing, rules, and frame of the table.

```
    style table from table /
       cellspacing=0 rules=groups frame=void;
    end;
run;
```

Edit the Base.Datasets.Members table template. The EDIT statement, along with the table template DEFINE statements and attributes, modifies the Base.Datasets.Members table template. For more information about creating and modifying table templates, see Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593.

```
proc template;
   edit Base.Datasets.Members;
      header hd1;
      footer ft1;
      define hd1;
         preformatted=on;
         just=l;
         text"     Table Header with Leading and Trailing Blanks    ";
      end;
      define ft1;
         preformatted=on;
         just=l;
         text"     Table Footer with Leading and Trailing Blanks    ";
      end;
      edit name;
       define header myheader;
         just=l;
         preformatted=on;
         text "     My new header";
       end;
       header=myheader;
       width=memname_width width_max=memname_width_max;
       preformatted=on;
      end;
   end;
run;
```

Create the RTF file, select the output object and run PROC DATASETS. The ODS RTF statement specifies the file that will contain the RTF output. The STYLE= option specifies the style to apply to the output. The ODS SELECT statement selects the output object Members to be sent to the open destinations.

```
ods rtf file="headerfooters.rtf" style=HeadersFootersBorders;
ods select members;
proc datasets lib=nlits;
run;
quit;
```

Close the open destinations and open the LISTING destination. The ODS _ALL_ CLOSE statement closes all open destinations and the files that are associated with them. If you do not close the destinations, then you will not be able to view the files. The ODS LISTING statement opens the LISTING destination.

```
ods _all_ close;
ods listing;
```

RTF Output

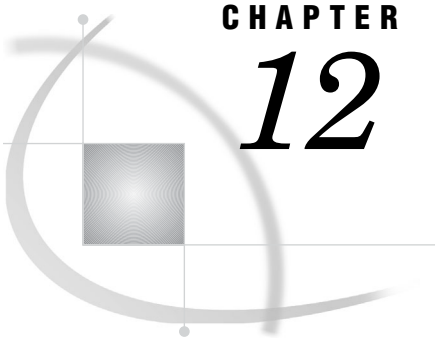
Display 11.18 RTF Output with Custom Headers and Footers

TableHeaderContainer, TableFooterContainer, and Border Control Style Attributes
Allows Control of Borders Between the Header, Body, and Footer of a Table

Table Header with Leading and Trailing Blanks

#	My new header	Member Type	File Size	Last Modified
1	Stats	DATA	5120	21Aug06:10:46:18
2	Stats2	DATA	5120	21Aug06:10:46:18

Table Footer with Leading and Trailing Blanks



CHAPTER

12

TEMPLATE Procedure: Creating Tabular Output

Overview: ODS Tabular Output

Using the TEMPLATE Procedure to Create or Customize Tabular Output

The TEMPLATE procedure enables you to customize the tabular appearance of your SAS output. With the TEMPLATE procedure, you can create and modify table templates, column templates, header templates, and footer templates. The Output Delivery System then uses these templates to produce customized tabular output for better data presentations and reports than what you get with the default SAS output. You can also create you own master tables using templates.

By default, ODS output is formatted according to the various definitions or templates that the procedure or DATA step specify. However, you can customize existing tabular output templates, or create your own new tabular output templates, by using the TEMPLATE procedure with these statements.

Table 12.1 PROC TEMPLATE Statements

Customization	Element Modified	Statement
Column presentation	Column template	DEFINE COLUMN
Table footer	Footer template	DEFINE FOOTER
Table header	Header template	DEFINE HEADER
Single output object	Table template	DEFINE TABLE
An existing template for a table, column, header, or footer	Table, column, header, footer	EDIT

Terminology

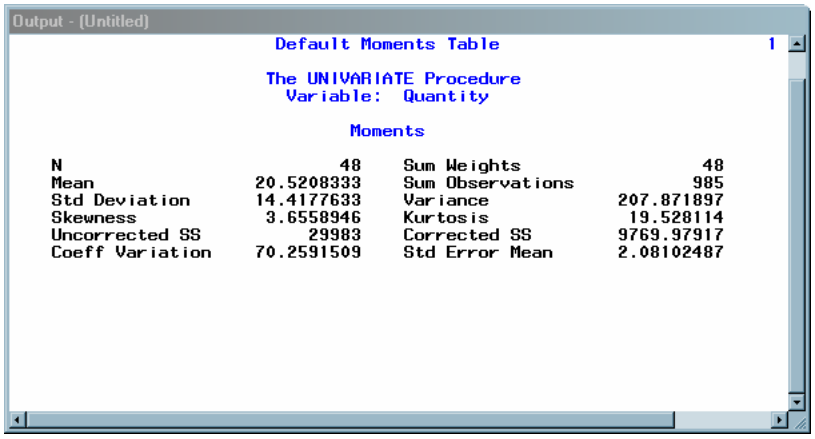
For definitions of terms used in this section, see "Terminology: TEMPLATE Procedure" on page 402.

What You Can Do With a Table Template

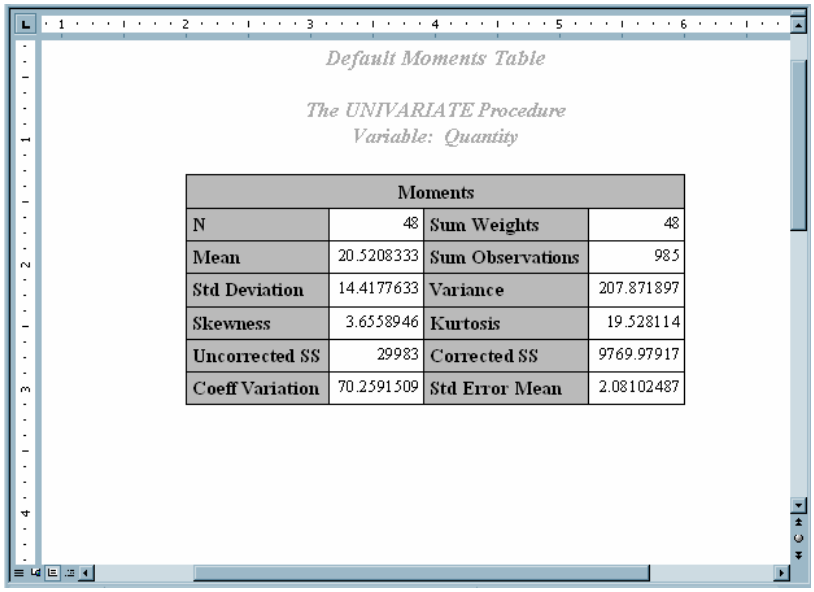
Default Listing and RTF Display of an Output Object

By default, ODS uses the table template specified by the procedure or DATA step to create ODS output. For example, the following display shows the default listing output of the Moments output object created by PROC UNIVARIATE. The second display shows the default RTF output of the same output object.

Display 12.1 Listing Output from PROC UNIVARIATE (Default Moments Table)



Display 12.2 RTF Output of Sales Statistics from PROC UNIVARIATE (Default Moments Table)



Default Moments Table

The UNIVARIATE Procedure
Variable: Quantity

Moments			
N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Customized Version of the Listing and RTF Display of an Output Object

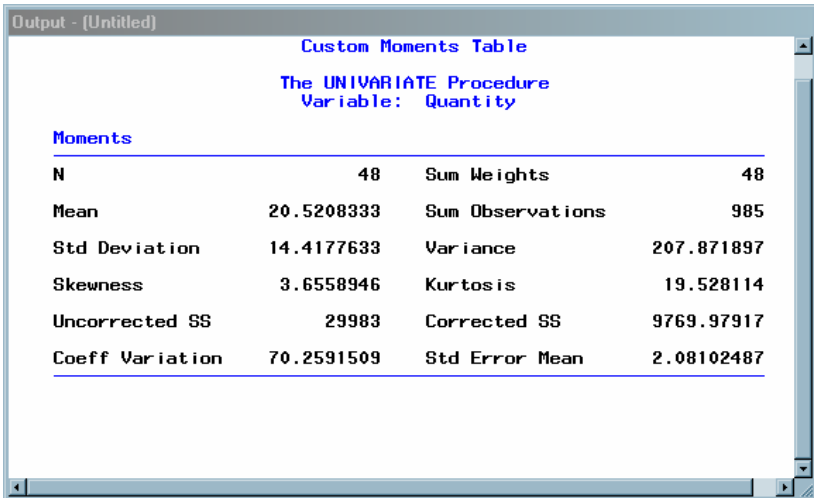
With PROC TEMPLATE, you can change many of the table elements and obtain a customized format for the output objects. Here are some of the elements that you can change:

- the color and the font of the text of the first table header
- the justification of the first table header
- the setting of the table attributes UNDERLINE and OVERLINE
- the line spacing between the rows

Note: Not all table template changes affect all destinations. For example, font changes are ignored in the LISTING destination. △

The following displays show the results of using a customized table template that changes the first table header attributes, sets underlining and overlining in the table, and changes the amount of spacing between rows.

Display 12.3 Listing Output from PROC UNIVARIATE (Customized Moments Table)



Output - (Untitled)

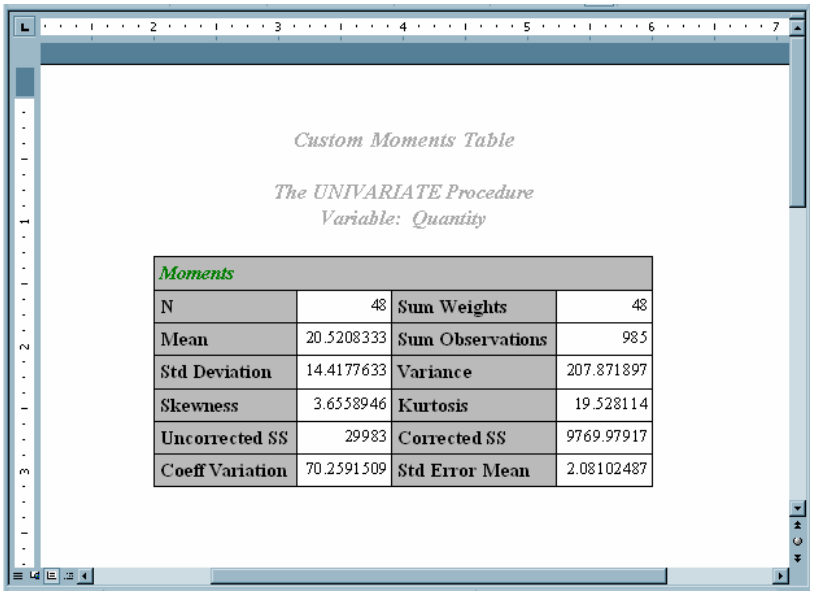
Custom Moments Table

The UNIVARIATE Procedure
Variable: Quantity

Moments

N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Display 12.4 RTF Output of Sales Statistics from PROC UNIVARIATE (Customized Moments Table)



Custom Moments Table

The UNIVARIATE Procedure
Variable: Quantity

Moments			
N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Comparing the Edit of an Existing Table Template with Creating a New Table Template

To change a table template without completely redefining it, use an EDIT statement. Using the EDIT statement keeps all of the templates and attributes that already exist in the table template, and changes only the templates or attributes specified in the EDIT statement. By default, the modified table template is stored in SASUSER.TEMPLAT with the same name as the table template specified in the EDIT statement.

To create a new table template, use the DEFINE TABLE statement. A table template cannot be a parent to itself because creating a table through inheritance causes an error, and then the template must be deleted. When you create a new table template, only the columns, headers, footers, and table attributes that you define exist in the new table template.

Note: If you edit an existing table or define a new table with the same name as an existing table, then the table template is stored in the SASUSER.TEMPLAT item store. This table template is used, by default, unless you specify that the Sashelp.Tmplmst path is searched first. However, you can use the ODS PATH statement to store the template elsewhere and access it differently. See the “ODS PATH Statement” on page 206 for more information. △

Tabular Syntax: TEMPLATE Procedure

PROC TEMPLATE;
EDIT *template-path-1* <AS *template-path-2*> < / STORE=*libref.template-store* > ;
statements-and-attributes
END;

DEFINE COLUMN *column-path* | **Base.Template.Column**
< / STORE=*libref.template-store*>;
statements-and-attributes
END;

DEFINE FOOTER *footer-path* | **Base.Template.Footer**
< / STORE=*libref.template-store*>;
statements-and-attributes
END;

DEFINE HEADER *template-name* | **Base.Template.Header**;
statements-and-attributes
END;

DEFINE TABLE *table-path* | **Base.Template.Table**
</ STORE=*libref.template-store*>;
statements-and-attributes
END;

This table lists the statements that add different features to tabular SAS output.

Table 12.2 PROC TEMPLATE Statements

Task	Statement
Edit an existing template for a table, column, header, or footer	EDIT
Create a column template	DEFINE COLUMN
Create a footer template	DEFINE FOOTER
Create a header template	DEFINE HEADER
Create a table template	DEFINE TABLE

EDIT Statement

Edits an existing template for a table, column, header, or footer.

Requirement: An END statement must follow the EDIT statement and all of the editing instructions.

Interaction: In some cases, you can use an EDIT statement inside a set of editing instructions.

When you edit a table template, you can also edit one or more column, header, or footer templates that are defined in the table.

When you edit a column template, you can also edit one or more header templates that are defined for that column.

Restriction: If you edit a template that is a link, the link is broken and a separate template is created.

Featured in: Example 1 on page 756 and Example 2 on page 762

EDIT *template-path-1* <**AS** *template-path-2* > </ STORE=*libref.template-store*>;
attribute-statements;

END;

Required Arguments

template-path-1
specifies a template to edit. *template-path-1* consists of one or more names that are separated by periods. Each name represents a directory in a template store, which is a type of SAS file.

Interaction: The STORE= option specifies a particular template store to read from and write to.

Tip: To determine the templates that a procedure or DATA step uses, submit the ODS TRACE ON statement before you run the SAS program. (See "ODS TRACE Statement" on page 317.)

Options

AS *template-path-2*
specifies the location in which to store the edited template, where *template-path-2* consists of one or more names that are separated by periods. Each name represents a directory in a template store, which is a type of SAS file. By default, PROC TEMPLATE writes the edited template to the first writable template store in the current path.

Default: If you omit **AS** *template-path-2*, PROC TEMPLATE writes the edited template to *template-path-1* in the first writable template store.

Restriction: If the current EDIT statement is inside a set of editing instructions, do not use the **AS** *template-path-2* option.

STORE=*libref.template-store*
specifies the template store from which to read *template-path-1* and in which to store *template-path-2*.

Statements and Attributes

The EDIT statement supports the same statements and attributes as the DEFINE TABLE statements. For more information, see "DEFINE TABLE Statement" on page 640.

Editing an Existing Template

There are two steps to follow to edit an existing template.

1 Open a copy of the specified file.

By default, PROC TEMPLATE looks for *template-path-1* in the list of template stores that is defined by the PATH statement. (See "PATH Statement" on page 416.) It opens a copy of the first template path that it finds in a template store that has Read access.

2 Save the modified file.

PROC TEMPLATE writes the modified template to the first template store in the current path with update access. If you omit a second template path to write to, then PROC TEMPLATE uses *template-path-1*. Therefore, if the template store from which *template-path-1* is read has Update access, you are actually modifying

the original template. Otherwise, the modified file is written to a template store to which you do have Update access.

If you do specify a second template path, then PROC TEMPLATE writes the edited template to the specified path in the first template store to which you have Write access.

DEFINE COLUMN Statement

Creates a template for a column.

Requirement: An END statement must be the last statement in the template.

Interaction: A column template can include one or more header templates.

See also: "DEFINE HEADER Statement" on page 626

Featured in: Example 3 on page 769 and Example 4 on page 777

DEFINE COLUMN *column-path* | **Base.Template.Column**
< / STORE=*libref.template-store*>;
<*column-attribute-1*; <*… column-attribute-n*; >>

CELLSTYLE *expression-1* **AS** <*style-element-name*><[*style-attribute-specification(s)*]><*…, expression-n* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>>;

COMPUTE AS *expression*;

DEFINE HEADER | **Base.Template.Header** *template-path*;
statements-and-attributes
END;

DYNAMIC *variable-1*<"*text-1*"> <*… variable-n*<"*text-n*">>;

MVAR *variable-1*<"*text-1*"> <*… variable-n*<"*text-n*">>;

NMVAR *variable-1*<"*text-1*"> <*… variable-n*<"*text-n*">>;

NOTES "*text*";

TRANSLATE *expression-1* **INTO** *expression-2* <*…, expression-n* **INTO** *expression-m*>;

END;

Table 12.3 DEFINE COLUMN Statements

Task	Statement
Set one or more column attributes	*column-attribute(s)*
Set the style element of the cells in the column according to the values of the variables	CELLSTYLE AS
Compute values for a column that is not in the data component, or modify the values of a column that is in the data component	COMPUTE AS
Create a template for a column header	DEFINE HEADER

Task	Statement
Define a symbol that references a value that the data component supplies from the procedure or DATA step	DYNAMIC
Define a symbol that references a macro variable. ODS will use the variable as a string. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.	MVAR
Define a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.	NMVAR
Provide information about the column	NOTES
Translate the specified values to other values	TRANSLATE INTO
End the template	END

Required Arguments

column-path
specifies where to store the column template. A *column-path* consists of one or more names that are separated by periods. Each name represents a directory in a template store, which is a type of SAS file. PROC TEMPLATE writes the template to the first writable template store in the current path.

Restriction: If the template is nested inside another template, *template-path* must be a single-level name because the nested template is stored in the same location as the original template.

Restriction: To reference the template that you are creating from another template, do not nest the template inside another one. For example, to reference a column template from multiple tables, do not define the column inside a table template.

Base.Template.Column
creates a master column template that is globally applied to all of your tabular output. After you create this template, you do not need to specify it explicitly in your SAS programs. It is automatically applied to all tabular output until you specifically remove the template from the item store.

Interaction: The Base.Template.Column master template attributes are overridden by other table templates.

Featured in: Example 6 on page 788

Options

STORE=*libref.template-store*
specifies the template store in which to store the template. If the template store does not exist, it is created.

Restriction: If the template is nested inside another template, do not use the STORE= option for the nested template because it is stored in the same location as the original template.

Restriction: The STORE= option does not become part of the template.

Column Attributes

This section lists all of the attributes that you can use in a column template. For all of the attributes that support a value of ON, these forms are equivalent:

ATTRIBUTE-NAME
ATTRIBUTE-NAME=ON

For all of the attributes that support a value of *variable*, *variable* is any variable that you declare in the column template with the DYNAMIC, MVAR, or NMVAR statement. If the attribute is Boolean, then the value of *variable* should resolve to either true or false as shown in this table:

Table 12.4 Boolean Values

True	False
ON	OFF
_ON_	_OFF_
1	0
TRUE	FALSE
YES	NO
YES	_NO_

Table 12.5 Column Attributes

Task	Attribute	Destinations
Influence the appearance of the cells contents		
Specify whether to suppress the value of a variable from one row to the next, if the value does not change based on the formatted value of the variable	BLANK_DUPS	All except OUTPUT
Specify whether to suppress the value of a variable from one row to the next if the value does not change based on the raw value of the variable	BLANK_INTERNAL_DUPS	All except OUTPUT
Select the best format for a column of a table	CHOOSE_FORMAT=	All
Specify whether to wrap the text in the current column	FLOW	LISTING

Task	Attribute	Destinations
Specify the format for the column	FORMAT=	All
Specify the number of decimals for the column if it is not specified with FORMAT= column attribute	FORMAT_NDEC=	All
Specify the format width for the column if it is not specified with FORMAT= column attribute	FORMAT_WIDTH=	All
Supply a numeric value against which values in the column are compared to eliminate trivial values from printing	FUZZ=	All except OUTPUT
Specify the horizontal justification of the format field within the column (and for the column header if the template for the header does not include JUST=)	JUST=	All except OUTPUT
Specify whether to justify the format field within the column, or to justify the value within the column, without regard to the format field	JUSTIFY	All destinations except LISTING behave as if JUSTIFY=ON
When the text in the column uses more than one line, specify whether to try to divide the text equally among all lines or to maximize the amount of text in each line	MAXIMIZE	LISTING
Specify whether to draw a continuous line in the current column above the first table footer or below the last row of the column if there is no table footer	OVERLINE	LISTING
Specify whether to treat the text as preformatted text	PREFORMATTED	Markup family, printer family, and RTF
Specify whether to print the column	PRINT	All except OUTPUT
Specify a separator character to append to each value in the column	SEPARATOR=	LISTING
Specify the style element and style attributes to use for the column	STYLE=	Markup family, printer family, and RTF
Specify that the text graphic columns be turned off when a procedure is going to output a graph	TEXT_GRAPHIC=	All except OUTPUT and DOCUMENT
Specify the split character for the data in the column	TEXT_SPLIT=	All except OUTPUT
Specify whether to draw a continuous line in the current column below the column header, or above the first row of the column if there is no column header	UNDERLINE=	LISTING

Task	Attribute	Destinations
Specify the vertical justification for the column	VJUST=	Markup family, printer family, and RTF
Specify the width of the column in characters	WIDTH=	LISTING
Specify the maximum width for this column	WIDTH_MAX=	LISTING
Customize column headers		
Specify the text for the column header or the name of the header template	HEADER=	All
Specify whether to print the column header	PRINT_HEADERS	All except OUTPUT
Influence the relationship to other columns		
Specify whether the column template is generic; that is, whether more than one variable use the template	GENERIC=	All except OUTPUT
Specify whether the column is an ID column	ID	LISTING and printer family
Specify whether to merge the current column with the column immediately to its right	MERGE	All except OUTPUT
Specify whether to merge the current column with the column immediately to its left	PRE_MERGE	All except OUTPUT
Specify the number of blank characters to leave between the current column and the column immediately to its left	PRE_SPACE=	LISTING
Specify the number of blank characters to leave between the current column and the column immediately to its right	SPACE=	LISTING
Influence the presentation of data panels		
Influence the place at which ODS splits a table when it creates multiple data panels	GLUE=	LISTING, printer family, and RTF
Specify whether to delete the current column from the output object if doing so enables all the remaining columns to fit in the space that is provided without splitting the table into multiple data panels	OPTIONAL	LISTING
Other column attributes		

Task	Attribute	Destinations
Specify which format to use if both a column template and a data component specify a format	DATA_FORMAT_OVERRIDE	All
Specify the name of the column in the data component to associate with the current column	DATANAME=	All
Specify which special characters in headers for generic columns are to be used as split characters	DEF_SPLIT	All
Specify whether to include the column in an output data set	DROP	OUTPUT
Specify a label for the column	LABEL=	OUTPUT
Specify the column template that the current template inherits from	PARENT=	All
Specify the name to use for the corresponding variable in an output data set	VARNAME=	OUTPUT

BLANK_DUPS<=ON | OFF | *variable*>
specifies whether to suppress the value of a variable from one row to the next if the value does not change according to the formatted value of the variable.

Default: OFF

Interaction: If the CLASSLEVELS= table attribute on page 646 is in effect, ODS ignores BLANK_DUPS=ON when any value changes in a preceding column that is also marked with BLANK_DUPS=ON.

Tip: The BLANK_DUPS attribute is valid in all destinations except the OUTPUT destination.

Note: When the PRINTER destination suppresses the value of a variable, it also suppresses the horizontal rule above the blank cell. △

Featured in: Example 4 on page 777

BLANK_INTERNAL_DUPS<=ON | OFF | *variable*>
specifies whether to suppress the value of a variable from one row to the next if the value does not change according to the raw value of the variable.

Default: OFF

Interaction: If the CLASSLEVELS= table attribute on page 646 is in effect, ODS ignores BLANK_INTERNAL_DUPS=ON when any value changes in a preceding column that is also marked with BLANK_INTERNAL_DUPS=ON.

Tip: The BLANK_INTERNAL_DUPS attribute is valid in all destinations except the OUTPUT destination.

Note: When the PRINTER destination suppresses the value of a variable, it also suppresses the horizontal rule above the blank cell. △

CHOOSE_FORMAT= COMPROMISE | MAX | MAX_ABS | MIN_MAX
selects a format based on the actual values in the column of the table.

Default: If you omit the CHOOSE_FORMAT column attribute, then the default format is determined by either the data component or other attributes.

Restriction: CHOOSE_FORMAT is not supported for computed columns because those columns' values are computed outside of the data object.

Tip: If you specify a small value for the FORMAT_WIDTH= option, then CHOOSE_FORMAT might create a dw.3 format.

Tip: The CHOOSE_FORMAT= attribute is valid in all destinations.

See: "Formatting Values in Table Columns" on page 755 for more information about column formats

COMPROMISE
looks at all of the values in the column and selects a good compromise format that works well for most values, but extreme values might shift to BEST format.

Tip: FORMAT_NDEC=d specifies the precision in digits.

Tip: The FORMAT_WIDTH= option suggests a maximum width. The actual format width might be smaller or it might be larger.

MAX
selects a format based on the maximum value in the column. Values are all expected to be positive so no space is reserved for a minus sign.

Default: By default, FORMAT_WIDTH=10 and FORMAT_NDEC= is ignored.

MAX_ABS
selects a format based on the maximum absolute value in the column. The format reserves space for a minus sign whether it is needed or not.

MIN_MAX
selects a format based on the minimum and maximum value in the column. The format reserves space for a minus sign only where it is actually needed.

Interaction: If FORMAT_NDEC=*d* is specified, a maximum of *d* decimal places is used.

DATA_FORMAT_OVERRIDE<=ON | OFF | *variable*>
specifies which format to use if both a column template and a data component specify a format.

Default: OFF

Tip: The DATA_FORMAT_OVERRIDE attribute is valid in all destinations.

ON
uses the format in the data component.

OFF
uses the format in the column template.

variable
uses the format of the specified variable.

DATANAME=*column-name*
specifies the name of the column in the data component to associate with the current column.

Default: By default, ODS associates the current column with a column of the same name in the data component.

Tip: The DATANAME= attribute is valid in all destinations.

DEF_SPLIT
specifies which special characters in headers for generic columns are to be used as split characters.

Tip: The DEF_SPLIT destination is valid in all destinations.

DROP<=ON | OFF | *variable*>
specifies whether to include the column in an output data set.

Default: OFF

Tip: The DROP attribute is valid only in the OUTPUT destination.

FLOW<=ON | OFF | *variable*>
specifies whether to wrap the text in the current column if it is too long to fit in the space that is provided.

Default: ON if the format width of the column is greater than the column width. OFF if the format width of the column is not greater than the column width.

See also: MAXIMIZE= on page 609

Tip: The FLOW attribute is valid only in the LISTING destination.

Note: The HTML and PRINTER destinations always wrap the text if it is too long to fit in the space that is provided. △

FORMAT=*format-name <format-width <decimal-width>>* | *variable*
specifies the format for the column.

Default: If you omit the FORMAT= option, then the format that the data component provides is used. If the data component does not provide a format, ODS uses one of the following:

- BEST8. for integers
- D12.3 for doubles
- the length of the variable for character variables

Restriction: If you specify a format width for a numeric column, then its value cannot exceed 32.

Tip: The FORMAT= attribute is valid in all destinations.

FORMAT_NDEC= *number* | *variable*
specifies the number of decimals for the column.

Default: The decimal width that is specified with the FORMAT= column attribute

Range: Number is a whole number from 0 to 32

Interaction: If you specify a decimal width using both the FORMAT= and the FORMAT_NDEC= attributes, then ODS uses the width that you specify with the FORMAT= attribute.

Tip: The FORMAT_NDEC= attribute is valid in all attributes.

FORMAT_WIDTH=*positive-integer* | *variable*
specifies the format width for the column.

Default: If you omit the column attribute FORMAT_WIDTH=, then ODS uses the format specified in the FORMAT= attribute.

Range: 1 to 32 for numeric variables; operating system limit for character variables

Interaction: If you specify a format width using both the FORMAT= and the FORMAT_WIDTH= attributes, then the width that you specify with the FORMAT= attribute is used.

Tip: The FORMAT_WIDTH= attribute is valid in all destinations.

FUZZ=*number* | *variable*
supplies a numeric value against which to compare values in the column to eliminate trivial values from printing. A number whose absolute value is less than or equal to the FUZZ= value is printed as 0. However, the real value of the number is used in any computations based on that number.

Default: This is the smallest representable floating-point number on the computer that you are using.

Tip: The FUZZ= attribute is valid in all destinations except the OUTPUT destination.

GENERIC<=ON | OFF | *variable*>
specifies whether the column template can be used by more than one column. Generic columns are useful in tables with many similar columns. For example, the table templates for both PROC SQL and the DATA step define only two columns: one for character variables and one for numeric variables. When a program runs, it determines which column template the data component should use for each column.

Default: OFF

Tip: The GENERIC attribute is valid in all destinations except the OUTPUT destination.

Featured in: Example 3 on page 769 and Example 4 on page 777

GLUE=*integer* | *variable*
Influences the places at which ODS splits a table when it creates multiple data panels. ODS creates multiple data panels from a table that is too wide to fit in the allotted space. The higher the value of GLUE= is, the less likely it is that ODS will split the table between the current column and the column to its right.

Default: 1

Range: -1 to 327

Tip: A value of -1 forces the table to split between the current column and the column to its right.

Tip: The GLUE= attribute is valid only in the LISTING, printer family, and RTF destinations.

HEADER=*header-specification*
specifies the text for the column header or the name of the header template. *header-specification* is one of the following:

"*text*"
specifies the actual text of the header.

Requirement: *text* must be enclosed by quotation marks.

header-name
specifies the name of a header template to use. Create a header template with the DEFINE HEADER statement (see "DEFINE HEADER Statement" on page 626). If *header-name* is a single-level name, the header template must occur within the current column template.

variable
specifies the name of a variable declared with the DYNAMIC, MVAR, or NMVAR statement. The value of the variable becomes the column header.

LABEL
Uses the label that is specified in the data component for the column header.

Default: _LABEL_

Interaction: If you are using the OUTPUT destination, then the HEADER= attribute does not change the label of the variable in the data set. To change the label in the data set, use the LABEL= attribute.

Tip: The HEADER= option provides a simple way to specify the text of a column header. To customize the header further, use the DEFINE HEADER statement with the appropriate header attributes. (See "DEFINE HEADER Statement" on page 626.)

Tip: Use the split character in the text of the header to force the text to a new line.

See also: LABEL= on page 609 and TEXT_SPLIT= on page 613

Tip: The HEADER= attribute is valid in all destinations.

Featured in: Example 3 on page 769 and Example 1 on page 551

ID<=ON | OFF | *variable*>
specifies whether the column is an ID column. An ID column is repeated on each data panel. (ODS creates multiple data panels when a table is too wide to fit in the allotted space.)

Default: OFF

Tip: ODS treats all columns up to and including a column that is marked with ID=ON as ID columns.

Tip: The ID attribute is valid only in LISTING and printer family destinations.

Featured in: Example 3 on page 769

JUST=*justification* | *variable*
specifies the horizontal justification of the format field within the column (and of the header if the template for the header does not include JUST=).
justification is one of the following:

CENTER
specifies center justification.

Alias: C

Interaction: To use center justification in printer family and RTF destinations, also specify JUSTIFY=ON.

DEC
specifies aligning the values by the decimal point.

Alias: D

Restriction: Decimal alignment is supported for printer family and RTF destinations only.

LEFT
specifies left justification.

Alias: L

RIGHT
specifies right justification.

Alias: R

Default: LEFT for columns that contain character values; RIGHT for columns that contain numeric values.

Interaction: The TEXTALIGN= style attribute overrides the value of JUST=.

Interaction: For the LISTING destination, ODS justifies the format field within the column width. At times, you can specify the JUSTIFY= attribute to get the results that you want. See the discussion of the JUSTIFY attribute on page 608.

Tip: The JUST= attribute is valid in all destinations except the OUTPUT destination.

Main discussion: "Values in Table Columns and How They Are Justified" on page 754

See also: FORMAT= on page 606 and WIDTH= on page 613

Featured in: Example 1 on page 756

JUSTIFY<=ON | OFF | *variable*>
specifies whether to justify the format field within the column or to justify the value within the column without regard to the format field.

Default: OFF

Interaction: JUSTIFY=ON can interfere with decimal alignment in the LISTING destination.

Tip: If you translate numeric data to character data, you might need to use JUSTIFY= to align the data.

Tip: All destinations except the LISTING destinations justify values as if JUSTIFY=ON.

Main discussion: "Values in Table Columns and How They Are Justified" on page 754

Featured in: Example 4 on page 777

LABEL="*text*" | *variable*
specifies a label for the column in the output data set.

Default: If you omit a label, ODS uses the label that is specified in the data component. If no label is specified in the data component, ODS uses the header for the column as the label.

Tip: The LABEL= attribute is valid only in the OUTPUT destination.

Tip: If the OUTPUT destination is open, then the LABEL= attribute provides a label for the corresponding variable in the output data set. This label overrides any label that is specified in the data component.

MAXIMIZE<=ON | OFF | *variable*>
specifies whether to try to divide the text equally among all lines or to maximize the amount of text in each line when the text in the column uses more than one line. For example, if the text spans three lines, MAXIMIZE=ON can result in 45% of the text on the first line, 45% of the text on the second line, and 10% of the text on the third line. MAXIMIZE=OFF can result in 33% of the text on each line. MAXIMIZE=ON can write lines of text that vary greatly in length. MAXIMIZE=OFF can result in using less than the full column width.

Default: OFF

Interaction: This attribute is effective only if the column is defined with FLOW=ON (see the discussion of the FLOW= attribute on page 606).

Tip: The MAXIMIZE= attribute is valid only in the LISTING destination.

MERGE<=ON | OFF | *variable*>
specifies whether to merge the current column with the column immediately to its right. When you set MERGE=ON for the current column, the data in each row of the column is merged with the data in the same row of the next column. ODS applies the format, justification, spacing, and prespacing attributes to each column independently. Then, it concatenates the columns. Finally, it applies to the concatenated data all the remaining attributes that are specified on the column that does not have MERGE= set.

Default: OFF

Restriction: You cannot use both MERGE=ON and PRE_MERGE=ON in the same column template. You cannot merge or premerge a column with another column that has either MERGE=ON or PRE_MERGE=ON. Note that you can merge three columns by setting MERGE=ON for the first column, no merge or premerge attributes for the second column, and PRE_MERGE=ON for the third column.

Tip: The MERGE= attribute is valid in all destinations except the OUTPUT destination.

See also: The PRE_MERGE= attribute on page 610

OPTIONAL<=ON | OFF | *variable*>
specifies whether to delete the current column from the output object if doing so enables all the remaining columns to fit in the space that is provided without splitting the table into multiple data panels.

Default: OFF

Interaction: If multiple column templates contain OPTIONAL=ON, either all or none of these columns are included in the output object.

Tip: The OPTIONAL attribute is valid only in the LISTING destination.

OVERLINE<=ON | OFF | *variable*>
specifies whether to draw a continuous line in the current column above the first table footer (or, if there is no table footer, below the last row of the column). The second formatting character is used to draw the line.

Default: OFF

Tip: The OVERLINE= attribute is valid only in the LISTING destination.

See also: For information on formatting characters see the discussion of the FORMCHAR= attribute on page 647.

PARENT=*variable*
specifies the column template from which the current template inherits attributes and statements. A *column-path* consists of one or more names that are separated by periods. Each name represents a directory in a template store, which is a type of SAS file. The current template inherits from the specified column in the first readable template store in the current path.

When you specify a parent, all of the attributes and statements that are specified in the parent's template are used in the current template unless the current template specifically overrides them.

Tip: The PARENT= attribute is valid in all destinations.

PREFORMATTED<=ON | OFF | *variable*>
specifies whether to treat the text as preformatted text. When text is preformatted, ODS honors line breaks as well as leading, trailing, and internal spaces. It also displays the text in a monospace font.

Default: OFF

Interaction: When PREFORMATTED=ON, ODS uses the DataFixed style element unless you specify another style element with the STYLE= column attribute.

Tip: The PREFORMATTED attribute is valid in the markup family, printer family, and RTF destinations.

PRE_MERGE<=ON | OFF | *variable*>
specifies whether to merge the current column with the column immediately to its left. When you set PRE_MERGE=ON for the current column, the data in each row of the column is merged with the data in the same row of the previous column. ODS applies the format, justification, spacing, and prespacing attributes to each column independently. Then, it concatenates the columns. Finally, it applies to the concatenated data all the remaining attributes that are specified on the column that does not have PRE_MERGE= set.

Default: OFF

Restriction: You cannot use both MERGE=ON and PRE_MERGE=ON in the same column template. You cannot merge or premerge a column with another column that has either MERGE=ON or PRE_MERGE=ON. Note that you can merge three columns by setting MERGE=ON for the first column, no merge or premerge attributes for the second column, and PRE_MERGE=ON for the third column.

Tip: The PRE_IMAGE attribute is valid in all destinations except the OUTPUT destination.

See also: MERGE= on page 609

PRE_SPACE=*non-negative-integer*
specifies the number of blank characters to leave between the current column and the column immediately to its left.

Default: A value in the range that is bounded by the COL_SPACE_MIN and COL_SPACE_MAX table attributes.

Interaction: If PRE_SPACE= and SPACE= are specified for the same intercolumn space, ODS honors PRE_SPACE=.

See also: The SPACE= column attribute on page 611, the COL_SPACE_MIN= table attribute on page 646, and the COL_SPACE_MAX= table attribute on page 646

Tip: The PRE_SPACE= attribute is valid only in the LISTING destination.

PRINT<=ON | OFF | *variable*>
specifies whether to print the column.

Default: ON

Interaction: If you specify the column attribute PRINT=OFF, then you turn off the value of a column if it is part of a stacked column. If all columns in a stacked column have PRINT=OFF set, then the entire column is removed from the table.

Tip: If all columns in a stacked column have PRINT=OFF specified, then the entire column is removed from the table.

Tip: The PRINT attribute is valid in all destination except the OUTPUT destination.

See also: The OPTIONAL= column attribute on page 610 and DROP= column attribute on page 605

PRINT_HEADERS<=ON | OFF | *variable*>
specifies whether to print the column header and any underlining and overlining.

Default: ON

See also: UNDERLINE= on page 613 and OVERLINE= on page 610

Tip: The PRINT_HEADERS attribute is valid in all destination except the OUTPUT destination.

SEPARATOR="*character*" | *variable*
specifies a separator character to append to each value in the column.

Default: None

Restriction: The SEPARATOR= column attribute is valid only for character variables.

Tip: To specify a hexadecimal character as the separator character, put an x after the closing quote. For example, this option assigns the hexadecimal character 2D as the separator character:

```
separator="2D"x
```

Tip: The SEPARATOR= attribute is valid only in the LISTING destination.

SPACE=*positive-integer* | *variable*
specifies the number of blank characters to leave between the current column and the column immediately to its right.

Default: A value in the range that is bounded by the COL_SPACE_MIN and COL_SPACE_MAX table attributes.

Interaction: If PRE_SPACE= and SPACE= are specified for the same intercolumn space, ODS honors PRE_SPACE=.

See also: The PRE_SPACE= column attribute on page 611, the COL_SPACE_MIN= table attribute on page 646, and the COL_SPACE_MAX= table attribute on page 646

Tip: The SPACE= attribute is valid only in the LISTING destination.

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the current column. Neither *style-attribute-specification* nor *style-element-name* is required. However, you must use at least one of them.

Note: You can use braces ({ and }) instead of square brackets ([and]). △

style-element-name
is the name of the style element to use to display the data in the column. The style element must be part of a style that is registered with the Output Delivery System. SAS provides some styles. You can create customized styles with PROC TEMPLATE (see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487). By default, ODS displays different parts of ODS output with different style elements. For example, by default, the data cells in a column are displayed with the style element Data. You would be most likely to use the following style elements with the STYLE= column attribute:

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataeEphasisFixed
- DataStrong
- DataStrongFixed

The style element provides the basis for displaying the column. Additional style attributes that you provide can modify the display.

For information on viewing a style so that you can see the style elements that are available, see "Viewing the Contents of a Style" on page 538. For information about the default style that ODS uses, see "Working with Styles" on page 538.

style-element-name is either the name of a style element or a variable whose value is a style element.

Default: Data

style-attribute-specification
describes the style attribute to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

For information on the style attributes that you can specify, see "Style Attributes and Their Values" on page 498.

Tip: The STYLE= attribute is valid only in the markup family, printer family, and RTF destinations.

Tip: If you use the STYLE= attribute inside a quoted string, then add a space before or after the carriage return to prevent errors. SAS does not interpret a carriage return as a space. You must explicitly specify spaces in quoted strings.

Featured in: Example 3 on page 769

TEXT_GRAPHIC= ON | OFF
specifies that the text graphic columns be turned off or on when a procedure is going to output a graph.

Default: OFF

TEXT_SPLIT="*character*" | *variable*
specifies the split character for the data in the column. The value in the column is broken when it reaches that character and continues the value on the next line. The split character itself is not part of the data and does not appear in the column.

Default: None

Tip: The TEXT_SPLIT= attribute is valid in all destinations except the OUTPUT destination.

UNDERLINE<=ON | OFF | *variable*>
specifies whether to draw a continuous line in the current column below the column header (or, if there is no column header, above the first row of the column). The second formatting character is used to draw the line.

Default: OFF

Main discussion: See the discussion of the FORMCHAR= attribute on page 647

Tip: The UNDERLINE= attribute is valid only in the LISTING destination.

VARNAME=*variable-name* | *variable*
specifies the name to use for the corresponding variable in an output data set.

Default: If you omit VARNAME=, then the value of the DATANAME= attribute is used. If you omit DATANAME=, then the name of the column is used.

Tip: If you use VARNAME= to specify the same name for different columns, a number is appended to the name each time that the name is used.

Tip: The VARNAME= attribute is valid only in the OUTPUT destination.

VJUST=*justification* | *variable*
Specifies the vertical justification for the column. *justification* is one of the following:

TOP
places the first line of text as high as possible.

Alias: T

CENTER
centers the text vertically.

Alias: C

BOTTOM
places the last line of text as low as possible.

Alias: B

Default: TOP

Tip: The VJUST= attribute is valid only in the markup family, printer family, and RTF destinations.

Featured in: Example 3 on page 769

WIDTH=*positive-integer* | *variable*
specifies the width of the column in characters.

Default: If you omit a width, the format width is used. If the column has no format associated with it, ODS uses one of the following widths:

- 8 for integers
- 12 for doubles

- data length for character variables

Interaction: The length of the column header can influence the width of the column.

See also: WIDTH_MAX= header attribute on page 614 and WIDTH= header attribute on page 637

Tip: The WIDTH= attribute is valid only in the LISTING destination.

WIDTH_MAX=*positive-integer* | *variable*
specifies the maximum width allowed for this column. By default, PROC TEMPLATE extends the width of the column if the header is wider than the data. The width of the column can be anywhere between the values of WIDTH= and WIDTH_MAX=.

Default: The width of the format for the column

Tip: The WIDTH_MAX= attribute is valid only in the LISTING destination.

CELLSTYLE AS Statement

Sets the style element of the cells in the column according to the values of the variables. Use this statement to set the presentation characteristics, such as foreground color and font face, of individual cells.

Featured in: Example 4 on page 777

CELLSTYLE *expression-1* **AS** <*style-element-name*><[*style-attribute-specification(s)*]><…, *expression-n* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>>;

Required Arguments

expression
is an expression that is evaluated for each cell in the column. If *expression* resolves to TRUE (a non-zero value), the style element that is specified is used for the current cell. If *expression* is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

expression has this form:

expression-1 <comparison-operator expression-n>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

SAS function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

built-in variable
is a special kind of WHERE expression operand that helps you find common values in column templates. Built-in variables are one or more of the following:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data column name.

DATATYPE
is the data type of the column variable. The data type is either numeric ("num") or character ("char").

Example: The following CELLSTYLE AS statement specifies that column variables that are numeric have a red font color and column variables that are character have a blue font color:

```
cellstyle  _datatype_ = "num" as {color=red},
             _datatype_ = "char" as {color=blue};
```

LABEL
is a column label.

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style element name that is used for the column.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 12.6 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Tip: Using an expression of 1 as the last expression in the CELLSTYLE AS statement sets the style element for any cells that did not meet an earlier condition.

Featured in: Example 5 on page 782

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

Options

Note: Neither *style-attribute-specification* nor *style-element-name* is required. However, you must use at least one of them. △

style-attribute-specification
describes a style attribute to set. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

For information on the style attributes that you can set in a column template, see "Style Attributes and Their Values" on page 498.

Default: If you do not specify any style attributes to modify, ODS uses the unmodified *style-element-name*.

style-element-name
is the name of the style element that displays the data in the column. The style element must be part of a style that is registered with the Output Delivery System. SAS provides some styles. Create customized styles by using PROC TEMPLATE (see "DEFINE STYLE Statement" on page 490). By default, ODS displays different parts of ODS output with different style elements. For example, by default, the data cells in a column are displayed with the style element Data. The style elements that you would be most likely to use with the CELLSTYLE AS statement in a column template are the following.

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataEmphasisFixed
- DataStrong
- DataStrongFixed

The style element provides the basis for displaying the column. Additional style attributes that you provide can modify the display.

Default: Data

See also: "Viewing the Contents of a Style" on page 538

See also: "Working with Styles" on page 538

COMPUTE AS Statement

Computes values for a column that is not in the data component, or modifies the values of a column that is in the data component.

COMPUTE AS *expression*;

Required Arguments

expression
is an expression that assigns a value to each table cell in the column.
expression has this form:

expression-1 <comparison-operator expression-n>

expression
is an arithmetic or logical sequence of operators and operands. An operator is a symbol that requests a comparison, a logical operation, or an arithmetic calculation. An operand is one of the following:

constant
is a fixed value, such as the name of a column, or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

To reference another column in a COMPUTE AS statement, use the name of the column. In addition, if the column has values in the data component, you can reference the column itself in the expression.

For example, this DEFINE COLUMN block defines a column that contains the square root of the value in the column called Source:

```
define column sqroot;
   compute as sqrt(source);
   header="Square Root";
   format=6.4;
end;
```

function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

built-in variable
is a special kind of WHERE expression operand that helps you find common values in column templates. Built-in variables are one or more of the following:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data-column name.

LABEL
is a column label.

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style-element name.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or another variable.

Table 12.7 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Tip: The COMPUTE AS statement can alter values in an output object. None of the templates that SAS provides modifies any values. To determine if a template was provided by SAS, use the "ODS VERIFY Statement" on page 325. If the template is not from SAS, the ODS VERIFY statement returns a warning when it runs the SAS program that uses the template. If you receive such a warning, use the SOURCE statement to look at the template and determine if the COMPUTE AS statement alters values. (See "SOURCE Statement" on page 417.)

Featured in: Example 5 on page 782

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

DEFINE HEADER Statement

Creates a template for a header inside a column template.

Main discussion: "DEFINE HEADER Statement" on page 626

DEFINE HEADERBase.Template.Header | *template-name*
</ STORE=*libref.template-store*>;
statements-and-attributes
END;

Required Arguments

template-name
specifies the name of a new header.

Restriction: *template-name* must be a single-level name.

Note: To reference the header template that you are creating from another template, create it outside of the column template. △

Base.Template.Header
creates a master header template that is globally applied to all of your tabular output. Once this template is created, you do not need to explicitly specify it in your SAS programs. The template is applied automatically to all tabular output until you specifically remove the template from the item store.

Interaction: The Base.Template.Header master template attributes are overridden by other table templates.

Featured in: Example 6 on page 788

statements-and-attributes
specifies the statements and header attributes that define a header inside a column.

See: "DEFINE HEADER Statement" on page 626

Options

STORE=*libref.template-store*
specifies the template store in which to store the template. If the template store does not exist, it is created.

Restriction: If the template is nested inside another template, do not use the STORE= option for the nested template because it is stored where the original template is stored.

Restriction: The STORE= option does not become part of the template.

DYNAMIC Statement

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Scope: You can use the DYNAMIC statement in the template of a table, column, header, or footer. A dynamic variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 1 on page 551 and Example 2 on page 557

DYNAMIC *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a variable that the data component supplies. ODS resolves the value of the variable when it binds the template and the data component.

Tip: Dynamic variables are most useful to the authors of SAS procedures and to DATA step programmers.

Options

text
is text that is placed in the template to explain the dynamic variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

MVAR Statement

Defines a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.

Scope: You can use the MVAR statement in the template of a table, column, header, or footer. A macro variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 3 on page 769 and Example 1 on page 551

MVAR *variable-1* <*'text-1'*> <*… variable-n* <*'text-n'*>>;

Required Arguments

variable
names a macro variable to reference in the template. ODS will use the value of the macro variable as a string. ODS does not resolve the value of the macro variable until it binds the template and the data component.

Tip: Declare macro variables this way in a template. For example, to use the automatic macro variable SYSDATE9 in a template, declare it in an MVAR statement and reference it as SYSDATE9, without an ampersand, in the PROC TEMPLATE step. If you use the ampersand, the macro variable resolves when the template is compiled instead of when ODS binds the template to the data component.

Options

text

is text that is placed in the template to explain the macro variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

NMVAR Statement

Defines a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.

Scope: You can use the NMVAR statement in the template of a table, column, header, or footer. A macro variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 4 on page 777

NMVAR *variable-1* <*'text-1'*> <*… variable-n* <*'text-n'*>>;

Required Arguments

variable
names a macro variable to reference in the template. ODS will convert the variable's value to a number (stored as a double) before using it. ODS does not resolve the macro variable until it binds the template and the data component.

Tip: Declare macro variables this way in a template. For example, to use a macro variable as a number, declare it in an NMVAR statement and reference it without an ampersand. If you use the ampersand, the macro variable resolves when the template is compiled instead of when ODS binds the template to the data component.

Options

text
is text that is placed in the template to explain the macro variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES Statement

Provides information about the table, header, column, or footer.

Tip: The NOTES statement becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

Featured in: Example 4 on page 777

NOTES *'text'*;

Required Arguments

text
provides information about the table.

TRANSLATE INTO Statement

Translates the specified values to other values.

TRANSLATE *expression-1* **INTO** *expression-2* <…,*expression-n* **INTO** *expression-m*>;

Required Arguments

expression-1
is an expression that is evaluated for each table cell in the column. If *expression-1* resolves to TRUE (a non-zero value), the translation that is specified is used for the current cell. If *expression-1* is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

expression has this form:

expression-1 <*comparison-operator expression-n*>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

SAS function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

built-in variable
is a special kind of WHERE expression operand that helps you find common values in column templates. Built-in variables are one or more of the following:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data column name.

DATATYPE
is the data type of the column variable. The data type is either numeric ("num") or character ("char").

LABEL
is a column label.

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style element name.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or another variable.

Table 12.8 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Tip: Using an expression of 1 as the last expression in the TRANSLATE–INTO statement specifies a translation for any cells that did not meet an earlier condition.

Featured in: Example 5 on page 782

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

expression-2
is an expression that specifies the value to use in the cell in place of the variable's actual value.

expression has this form:

expression-1 <comparison-operator expression-n>

expression

is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

SAS function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

Built-in variable
a special kind of WHERE expression operand that helps you find common values in table templates. Built-in variables are one or more of the following variables:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data column name.

DATATYPE
is the data type of the column variable. The data type is either numeric ("num") or character ("char").

LABEL
is a column label.

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style element name.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 12.9 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than

Symbol	Mnemonic Equivalent	Definition
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Restriction: *expression-2* must resolve to a character value, not a numeric value.

Tip: When you translate a numeric value to a character value, the column template does not try to apply the numeric format that is associated with the column. Instead, it simply writes the character value into the format field, starting at the left. To right-justify the value, use the JUSTIFY=ON attribute.

Featured in: Example 5 on page 782

See also: JUSTIFY= on page 608

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

END Statement

Ends the table template, header template, column template, or footer template.

END;

DEFINE FOOTER Statement

Creates a template for a table footer.

Requirement: An END statement must be the last statement in the template.

Featured in: Example 3 on page 769 and Example 1 on page 551

See: "DEFINE HEADER Statement" on page 626

DEFINE FOOTER *footer-path* | **Base.Template.Footer**
< / STORE=*libref.template-store*>;
<*footer-attribute-1*; <…*footer-attribute-n*; >>
DYNAMIC *variable-1* <"*text-1*"> <… *variable-n* <"*text-n*">>;
MVAR *variable-1* <"*text-1*"> <… *variable-n* <"*text-n*">>;
NMVAR *variable-1* <"*text-1*"> <… *variable-n* <"*text-n*">>;
NOTES "*text*";
TEXT *footer-specification*;
TEXT2 *footer-specification*;
TEXT3 *footer-specification*;

END;

The substatements in the DEFINE FOOTER statements and the footer attributes are the same as the substatements in the DEFINE HEADER statement and the header attributes. For details about substatements and footer attributes, see “DEFINE HEADER Statement” on page 626.

DEFINE HEADER Statement

Creates a template for a table header.

Requirement: An END statement must be the last statement in the template.

Featured in: Example 3 on page 769

DEFINE HEADER *header-path* | **Base.Template.Header**

</ STORE=*libref.template-store*>;

<*header-attribute-1*; <*… header-attribute-n*; >>

DYNAMIC *variable-1* <"*text-1*"> <*… variable-n* <"*text-n*">>;

MVAR *variable-1* <"*text-1*"> <*… variable-n* <"*text-n*">>;

NMVAR *variable-1* <"*text-1*"> <*… variable-n* <"*text-n*">>;

NOTES "*text*";

TEXT *header-specification*;

TEXT2 *header-specification*;

TEXT3 *header-specification*;

END;

Table 12.10 DEFINE HEADER Statements

Task	Statement
Set one or more header attributes	*header-attribute(s)*
Define a symbol that references a value that the data component supplies from the procedure or DATA step	DYNAMIC
Define a symbol that references a macro variable. ODS will use the variable as a string. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.	MVAR

Task	Statement
Define a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.	NMVAR
Provide information about the column	NOTES
Specify the text of the header	TEXT
Specify an alternative header to use in the listing output if the header that is provided by the TEXT statement is too long	TEXT2
Specify an alternative header to use in the listing output if the header that is provided by the TEXT2 statement is too long	TEXT3
End the header template	END

Required Arguments

header-path
specifies where to store the header template. A *header-path* consists of one or more names, separated by periods. Each name represents a directory in a template store. (A template store is a type of SAS file.) PROC TEMPLATE writes the template to the first writable template store in the current path.

Restriction: If the template is nested inside of another template, *header-path* must be a single-level name.

Restriction: To reference the template that you are creating from another template, do not nest the template inside another template. For example, to reference a header template from multiple columns, do not define the header inside a column template.

Base.Template.Header | Base.Template.Footer
creates a master header template that is globally applied to all of your tabular output. Once this template is created, you do not need to explicitly specify it in your SAS programs. It is automatically applied to all tabular output until you specifically remove it from the item store.

Interaction: The Base.Template.Header or Base.Template.Footer master template attributes are overridden by other table templates.

Featured in: Example 6 on page 788

Options

STORE=*libref.template-store*
specifies the template store in which to store the template. If the template store does not exist, it is created.

Restriction: If the template is nested inside another template, do not use the STORE= option for the nested template because it is stored where the original template is stored.

Restriction: The STORE= option does not become part of the template.

Header Attributes

This section lists all the attributes that you can use in a header template. A column header spans a single column. A spanning header spans multiple columns. These two kinds of headers are defined in the same way except that a spanning header uses the START= or the END= attribute, or both.

For all attributes that support a value of ON, these forms are equivalent:

```
ATTRIBUTE-NAME
ATTRIBUTE-NAME=ON
```

For all of the attributes that support a value of *variable*, *variable* is any variable that you declare in the table template with the DYNAMIC, MVAR, or NMVAR statement. If the attribute is a boolean, then the value of *variable* should resolve to either true or false as shown in this table:

Table 12.11 Boolean Values

True	False
ON	OFF
_ON_	_OFF_
TRUE	FALSE
YES	NO
YES	_NO_

Table 12.12 Header Attributes

Task	Attribute	Destinations
Influence the appearance of the contents of the header		
Specify that special characters in headers for generic columns are to be used as split characters	DEF_SPLIT	All
Specify whether to try to expand the column width to accommodate the longest word in the column header	FORCE	LISTING
Specify the horizontal justification for the column header	JUST=	All except OUTPUT
Specify whether to try to divide the text equally among all lines or to maximize the amount of text in each line when the text in the header uses more than one line	MAXIMIZE	LISTING

Task	Attribute	Destinations
Specify whether to draw a continuous line above the header	OVERLINE	LISTING
Specify whether to treat the text as preformatted text	PREFORMATTED	Markup family, printer family, and RTF
Specify whether to print the header	PRINT	All
Specify the number of blank lines to place between the current header and the next header or between the current footer and the previous footer	SPACE=	LISTING
Specify the split character for the header	SPLIT=	All except OUTPUT
Specify the style element and any changes to its attributes to use for the header	STYLE=	Markup family, printer family, and RTF
Specify whether to start a new header line in the middle of a word	TRUNCATE	LISTING
Specify whether to draw a continuous line underneath the header	UNDERLINE	LISTING
Specify vertical justification for the header	VJUST=	Markup family, PRINTER, family, and RTF
Specify the width of the header in characters	WIDTH=	LISTING
Influence the content of the header		
Specify a character to use to expand the header to fill the space over the column or columns that the header spans	EXPAND=	LISTING
Specify whether to repeat the text of the header until the space that is allotted for the header is filled	REPEAT	LISTING
Influence the placement of the header		
Specify the last column that a spanning header covers	END=	All except OUTPUT
Specify the first column that a spanning header covers	START=	All except OUTPUT
Specify whether to expand the header to reach the sides of the page	EXPAND_PAGE	LISTING
Specify whether a spanning header appears only on the first data panel if the table is too wide to fit in the space that is provided	FIRST_PANEL	LISTING, printer family, and RTF

Task	Attribute	Destinations
Specify whether a table footer appears only on the last data panel if the table is too wide to fit in the space that is provided	LAST_PANEL	LISTING, printer family, and RTF
Specify whether to extend the text of the header into the header space of adjacent columns	SPILL_ADJ	LISTING
Specify whether to extend the text of the header into the adjacent margin	SPILL_MARGIN	LISTING
Other header attributes		
Specify an abbreviation for the header *	ABBR=	MARKUP
Specify an acronym for the header *	ACRONYM=	MARKUP
Specify an alternate description for the header *	ALT=	MARKUP
Specify whether multiple columns can use the header	GENERIC	All except OUTPUT
Specify a long description for the header *	LONGDESC=	MARKUP
Specify the header template that the current template inherits from	PARENT=	All

* SAS includes these accessibility and compatibility features that improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

ABBR= "*text*" | *variable*
specifies an abbreviation for the header.

Requirement: The text must be enclosed with quotation marks.

Tip: The ABBR attribute is valid only in the MARKUP destination.

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

ACRONYM= "*text*" | *variable*
specifies an acronym for the header.

Requirement: The text must be enclosed with quotation marks.

Tip: The ACRONYM= attribute is valid only in the MARKUP destination.

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

ALT= "*text*" | *variable*
specifies an alternate description of the header.

Requirement: The text must be enclosed with quotation marks.

Tip: The ALT= attribute is valid only in the MARKUP destination.

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

DEF_SPLIT
specifies which special characters in headers for generic columns are to be used as split characters.

Tip: The DEF_SPLIT attribute is valid in all destinations.

END=*column-name* | *variable*
specifies the last column that a spanning header covers.

Default: The last column

See also: START= on page 634

Tip: The END= attribute is valid in all destinations except the OUTPUT destination.

EXPAND="*string*" | *variable*
specifies a character to use to expand the header to fill the space over the column or columns that the header spans.

Default: None

Interaction: If you specify both the REPEAT=ON and EXPAND=ON attributes, then the EXPAND= attribute is used.

See also: REPEAT= on page 633

Tip: If the string or the variable that you specify contains more than one character, then only the first character is used.

Tip: The EXPAND= attribute is valid only in the LISTING destination.

See also: EXPAND_PAGE=

EXPAND_PAGE<= ON | OFF | *variable*>
specifies whether to expand the header to reach the sides of the page.

Default: OFF

See also: EXPAND=

Tip: The EXPAND_PAGE attribute is valid only in the LISTING destination.

FIRST_PANEL<= ON | OFF | *variable*>
specifies whether a spanning header appears only on the first data panel if the table is too wide to fit in the space that is provided.

Default: OFF

Restriction: Applies only to headers, not to footers

See also: LAST_PANEL= on page 632

Tip: The FIRST_PANEL attribute is valid in the LISTING, printer family, and RTF destinations.

FORCE<= ON | OFF | *variable*>
specifies whether to try to expand the column width to accommodate the longest word in the column header. The column width can be anything between the values for the WIDTH= and WIDTH_MAX= column attributes.

Default: ON

See also: WIDTH on page 613 and WIDTH_MAX on page 614

Tip: The FORCE= attribute is valid only in the LISTING destination.

GENERIC<= ON | OFF | *variable*>
specifies whether multiple columns can use the header.

Default: OFF

Restriction: This attribute is primarily for writers of SAS procedures and for DATA step programmers.

Tip: The GENERIC= attribute is valid in all destinations except the OUTPUT destination.

JUST=*justification* | *variable*
specifies the horizontal justification for the column header, where *justification* is one of the following:

LEFT
specifies left justification.

Alias: L

RIGHT
specifies right justification.

Alias: R

CENTER
specifies center justification.

Alias: C

Default: The justification for the column

Tip: The JUST= attribute is valid in all destinations except the OUTPUT destination.

Featured in: Example 1 on page 756

LAST_PANEL<= ON | OFF | *variable*>
specifies whether a table footer appears only on the last data panel if the table is too wide to fit in the space that is provided.

Default: OFF

Restriction: Applies only to footers, not to headers

See also: FIRST_PANEL on page 631

Tip: The LAST_PANEL= attribute is valid only in the LISTING, printer family, and RTF destinations.

LONGDESC= "*string*" | *variable*
specifies the long description of the header.

Requirement: The text must be enclosed within quotation marks.

Tip: The LONGDESC= attribute is valid only in markup family destinations.

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

MAXIMIZE<=ON | OFF | *variable*>
specifies whether to try to divide the text equally among all lines or to maximize the amount of text in each line when the text in the header uses more than one line. For example, if the text spans three lines, MAXIMIZE=ON can result in 45% of the text on the first line, 45% of the text on the second line, and 10% of the text on the third line. MAXIMIZE=OFF can result in 33% of the text on each line. MAXIMIZE=ON can write lines of text that vary greatly in length. MAXIMIZE=OFF can result in using less than the full column width.

Default: OFF

Tip: The MAXIMIZE= attribute is valid only in the LISTING destination.

OVERLINE<=ON | OFF | *variable*>
specifies whether to draw a continuous line above the header. The second formatting character is used to draw the line. (See the discussion of the FORMCHAR= attribute on page 647.)

Default: OFF

Tip: The OVERLINE= attribute is valid only in the LISTING destination.

PARENT=*header-path*
specifies the header template that the current template inherits from. A *header-path* consists of one or more names, separated by periods. Each name represents a directory in a template store. (A template store is a type of SAS file.) The current template inherits from the specified header template in the first readable template store in the current path.

When you specify a parent, all of the attributes and statements that are specified in the parent's template are used in the current template unless the current template specifically overrides them.

Tip: The PARENT= attribute is valid in all destinations.

PREFORMATTED<=ON | OFF | *variable*>
specifies whether to treat the text as preformatted text. When text is preformatted, ODS honors line breaks as well as leading, trailing, and internal spaces. It also displays the text in a monospace font.

Default: OFF

Interaction: When PREFORMATTED=ON, and you are defining a table header or a footer, ODS uses the HeaderFixed or the FooterFixed style element unless you specify another style element with the STYLE= column attribute.

When PREFORMATTED=ON, and you are defining a column header, ODS uses the RowHeaderFixed style element unless you specify another style element with the STYLE= column attribute.

Tip: The PREFORMATTED attribute is valid in the markup family, printer family, and RTF destinations.

PRINT<=ON | OFF | *variable*>
specifies whether to print the header.

Default: ON

Tip: When PRINT=ON, the column header becomes the label of the corresponding variable in any output data sets that the OUTPUT destination creates if neither the column template nor the data component provides a label.

Tip: The PRINT= attribute is valid in all destinations.

REPEAT<=ON | OFF | *variable*>
specifies whether to repeat the text of the header until the space that is allotted for the header is filled.

Default: OFF

Interaction: If you specify both the REPEAT=ON and EXPAND=ON attributes, then the EXPAND= attribute is used.

See also: EXPAND= on page 631

Tip: The REPEAT attribute is valid only in the LISTING destination.

SPACE=*positive-integer* | *variable*
specifies the number of blank lines to place between the current header and the next header or between the current footer and the previous footer.

Default: 0

Tip: A row of underlining or overlining is considered a header or a footer.

Tip: The SPACE= attribute is valid only in the LISTING destination.

Featured in: Example 3 on page 769

SPILL_ADJ<=ON | OFF | *variable*>
specifies whether to extend the text of the header into the header space of adjacent columns.

Default: OFF

Interaction: FORCE=, SPILL_MARGIN=, SPILL_ADJ=, and TRUNCATE= are mutually exclusive. If you specify more than one of these attributes, then only one of these attributes are used. FORCE= takes precedence over the other three attributes, followed by SPILL_MARGIN=, SPILL_ADJ=, and TRUNCATE=.

See also: The FORCE= header attribute on page 631, the SPILL_MARGIN= header attribute on page 634, and the TRUNCATE= header attribute on page 636

Tip: The SPILL_ADJ attribute is valid only in the LISTING destination.

SPILL_MARGIN<=ON | OFF | *variable*>
specifies whether to extend the text of the header into the adjacent margin.

Default: ON

Restriction: SPILL_MARGIN= applies only to a spanning header that spans all the columns in a data panel.

Interaction: The FORCE=, SPILL_MARGIN=, SPILL_ADJ=, and TRUNCATE= attributes are mutually exclusive. If you specify more than one of these attributes, then only one of these attributes are used. The FORCE= attribute takes precedence over the other three attributes, followed by SPILL_MARGIN=, SPILL_ADJ=, and TRUNCATE=.

See also: The FORCE= header attribute on page 631, the SPILL_ADJ header attribute on page 634, and the TRUNCATE= header attribute on page 636

Tip: The SPILL_MARGIN attribute is valid only in the LISTING destination.

SPLIT= "*character*" | *variable*
specifies the split character for the header. PROC TEMPLATE starts a new line when it reaches that character and continues the header on the next line. The split character itself is not part of the header although each occurrence of the split character counts toward the maximum length for a label.

Tip: The first character in a header is automatically treated as a split character if it is not one of the following:

- an alphanumeric character
- a blank
- an underscore (_)
- a hyphen (-).

Tip: The SPLIT= attribute is valid in all destinations except the OUTPUT destination.

START=*column-name* | *variable*
specifies the first column that a spanning header covers.

Default: The first column

See also: END on page 631

Tip: The START= attribute is valid in all destinations except the OUTPUT destination.

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the header.

style-element-name
is the name of the style element to use to produce the header. The style element must be part of a style that is registered with the Output Delivery System. SAS provides some styles. You can create customized styles by using PROC TEMPLATE (see "DEFINE STYLE Statement" on page 490). By default, ODS produces different parts of ODS output with different elements. For example, by default, a table header is displayed with the style element Header. The style elements that you would be most likely to use with the STYLE= attribute for a table header are as follows:

- Header
- HeaderFixed
- HeaderEmpty
- HeaderEmphasis
- HeaderEmphasisFixed
- HeaderStrong
- HeaderStrongFixed

The style elements that you would be most likely to use with the STYLE= attribute for a table footer are as follows:

- Footer
- FooterFixed
- FooterEmpty
- FooterEmphasis
- FooterEmphasisFixed
- FooterStrong
- FooterStrongFixed

The style elements that you would be most likely to use with the STYLE= attribute for a column header are as follows:

- Rowheader
- RowheaderFixed
- RowheaderEmpty
- RowheaderEmphasis
- RowheaderEmphasisFixed
- RowheaderStrong
- RowheaderStrongFixed

The style element provides the basis for displaying the header. Additional style attributes that you provide can modify the display.

style-element-name is either the name of a style element or a variable whose value is a style element.

Default: Header

See also: "Viewing the Contents of a Style" on page 538

See also: "Working with Styles" on page 538

style-attribute-specification

describes the style attribute to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

Requirement: The STYLE= option requires either a *style-attribute-specification* or a *style-element-name*.

Tip: You can use braces ({ and }) instead of square brackets ([and]).

Tip: If you use the STYLE= attribute inside a quoted string, then add a space before or after the carriage return to prevent errors. SAS does not interpret a carriage return as a space. You must explicitly specify spaces in quoted strings.

Tip: The STYLE= attribute is valid only in the markup family, printer family, and RTF destinations.

Featured in: Example 1 on page 756 and Example 3 on page 769

See also: "Style Attributes and Their Values" on page 498

TRUNCATE<=ON | OFF | *variable*>
specifies whether to start a new header line in the middle of a word.

ON
starts a new line of the header when the text fills the specified column width.

OFF
extends the width of the column to accommodate the longest word in the column header, if possible.

Note: TRUNCATE=OFF is the same as FORCE=ON. △

Default: OFF

Interaction: If you specify FORCE=, SPILL_MARGIN=, or SPILL_ADJ=, then the TRUNCATE= attribute is ignored.

See also: The FORCE= header attribute on page 631, the SPILL_MARGIN= header attribute on page 634, and the SPILL_ADJ header attribute on page 634

Tip: The TRUNCATE= attribute is valid only in the LISTING destination.

UNDERLINE<=ON | OFF | *variable*>
specifies whether to draw a continuous line below the header. The second formatting character is used to draw the line.

Default: OFF

Main discussion: See the discussion of the FORMCHAR= attribute on page 647.

Tip: The UNDERLINE attribute is valid only in the LISTING destination.

VJUST=*justification* | *variable*
Specifies vertical justification for the header. *justification* is one of the following:

TOP
places the header as high as possible.

Alias: T

CENTER
centers the header vertically.

Alias: C

BOTTOM
places the header as low as possible.

Alias: B

Default: BOTTOM

Tip: The VJUST= attribute is valid only in the MARKUP and PRINTER families of destinations.

WIDTH=*positive-integer* | *variable*
specifies the width of the header in characters.

Default: If you omit a width, the column width is used.

Tip: To create a vertical header, specify a width of 1.

Tip: The WIDTH= attribute is valid only in the LISTING destination.

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Scope: You can use the DYNAMIC statement in the template of a table, column, header, or footer. A dynamic variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 1 on page 551 and Example 2 on page 557

DYNAMIC *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a variable that the data component supplies. ODS resolves the value of the variable when it binds the template and the data component.

Tip: Dynamic variables are most useful to the authors of SAS procedures and to DATA step programmers.

Options

text
is text that is placed in the template to explain the dynamic variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

MVAR Statement

Defines a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.

Scope: You can use the MVAR statement in the template of a table, column, header, or footer. A macro variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 3 on page 769 and Example 1 on page 551

MVAR *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a macro variable to reference in the template. ODS will use the value of the macro variable as a string. ODS does not resolve the value of the macro variable until it binds the template and the data component.

Tip: Declare macro variables this way in a template. For example, to use the automatic macro variable SYSDATE9 in a template, declare it in an MVAR statement and reference it as SYSDATE9, without an ampersand, in the PROC TEMPLATE step. If you use the ampersand, the macro variable resolves when the template is compiled instead of when ODS binds the template to the data component.

Options

text
is text that is placed in the template to explain the macro variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

NMVAR Statement

Defines a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.

Scope: You can use the NMVAR statement in the template of a table, column, header, or footer. A macro variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 4 on page 777

NMVAR *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a macro variable to reference in the template. ODS will convert the variable's value to a number (stored as a double) before using it. ODS does not resolve the macro variable until it binds the template and the data component.

Tip: Declare macro variables this way in a template. For example, to use a macro variable as a number, declare it in an NMVAR statement and reference it without an ampersand. If you use the ampersand, the macro variable resolves when the

template is compiled instead of when ODS binds the template to the data component.

Options

text
is text that is placed in the template to explain the macro variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES Statement

Provides information about the table, header, column, or footer.

Tip: The NOTES statement becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

Featured in: Example 4 on page 777

NOTES '*text*';

Required Arguments

text
provides information about the table.

TEXT Statement

Specifies the text of the header or the label of a variable in an output data set.

Featured in: Example 3 on page 769

TEXT *header-specification(s)*;

Required Arguments

header-specification(s)
specifies the text of the header. Each *header-specification* is one of the following:

LABEL
uses the label of the object that the header applies to as the text of the header. For example, if the header is for a column, _LABEL_ specifies the label for the variable that is associated with the column. If the header is for a table, _LABEL_ specifies the label for the data set that is associated with the table.

text-specification(s)
specifies the text to use in the header. Each *text-specification* is one of the following:

- a quoted string
- a variable, followed by an optional format. The variable is any variable that is declared in a DYNAMIC, MVAR, or NMVAR statement.

Note: If the first character in a quoted string is neither a blank character nor an alphanumeric character, and SPLIT is not in effect, the TEXT statement treats that character as the split character. (See the discussion of SPLIT on page 634.) △

Default: If you omit a TEXT statement, the text of the header is the label of the object that the header applies to.

Tip: If the quoted string is a blank and it is the only item in the header specification, the header is a blank line.

Featured in: Example 3 on page 769

TEXT2 Statement

Provides an alternative header to use in the listing output if the header that is provided by the TEXT statement is too long.

See: "TEXT Statement" on page 639

TEXT3 Statement

Provides an alternative header to use in the listing output if the header that is provided by the TEXT2 statement is too long.

See: "TEXT Statement" on page 639

END Statement

Ends the table template, header template, column template, or footer template.

END;

DEFINE TABLE Statement

Creates a table template.

Requirement: An END statement must be the last statement in the template.

Interaction: A table template can contain one or more column, header, or footer templates.

Featured in: Example 3 on page 769 and Example 4 on page 777

DEFINE TABLE *table-path* | **Base.Template.Table**
</ STORE=*libref.template-store*>;
<*table-attribute-1*; <… *table-attribute-n*; >>
CELLSTYLE *expression-1* **AS** <*style-element-name*><[*style-attribute-specification(s)*]><…, *expression-n* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>>;
COLUMN *column(s)*;
DEFINE *template-type template-name* </ *option(s)*>;
statements-and-attributes
END;
DYNAMIC *variable-1* <"*text-1*"> <… *variable-n* <"*text-n*">>;
FOOTER *footer-name(s)*;
HEADER *header-name(s)*;
MVAR *variable-1* <"*text-1*"> <… *variable-n* <"*text-n*">>;
NMVAR *variable-1* <"*text-1*"> <… *variable-n* <"*text-n*">>;
NOTES "*text*";
TRANSLATE *expression-1* **INTO** *expression-2* <… , *expression-n* **INTO** *expression-m*;>
END;

Table 12.13 DEFINE TABLE Statements

Task	Statement
Set one or more table attributes	*table-attribute(s)*
Set the style element of the cells in the table that contain numeric variables according to the values of the variables	CELLSTYLE AS
Declare a symbol as a column in the table and specify the order of the columns	COLUMN
Create a template for a column, header, or footer	DEFINE
Define a symbol that references a value that the data component supplies from the procedure or DATA step	DYNAMIC
Declare a symbol as a footer in the table and specify the order of the footers	FOOTER
Declare a symbol as a header in the table and specify the order of the headers	HEADER
Define a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.	MVAR

Task	Statement
Define a symbol that references a macro variable. ODS will convert the value of the variable to a number (stored as a double) before use. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.	NMVAR
Provide information about the table	NOTES
Translate the specified numeric values to other values	TRANSLATE INTO
End a template, or end the editing of a template	END

Required Arguments

table-path
specifies where to store the table template. A *table-path* consists of one or more names that are separated by periods. Each name represents a directory in a template store, which is a type of SAS file. PROC TEMPLATE writes the template to the first writable template store in the current path.

Base.Template.Table
creates a master table template that is globally applied to all of your tabular output. Once this template is created, you do not need to explicitly specify it in your SAS programs. It is automatically applied to all tabular output until you specifically remove it from the item store.

Interaction: The Base.Template.Table master template attributes are overridden by other table templates.

Tip: The Base.Template.Table master template is most useful when used with the CELLSTYLE AS statements to create alternating colors in your tabular output.

Featured in: Example 6 on page 788

Options

STORE=*libref.template-store*
specifies the template store in which to store the template. If the template store does not exist, it is created.

Restriction: The STORE= option does not become part of the template.

Table Attributes

This section lists all the attributes that you can use in a table template. For all attributes that support a value of ON, these forms are equivalent:

```
ATTRIBUTE-NAME
ATTRIBUTE-NAME=ON
```

For all of the attributes that support a value of *variable*, *variable* is any variable that you declare in the table template with the DYNAMIC, MVAR, or NMVAR statement. If the attribute is a boolean, then the value of *variable* should resolve to either true or false as shown in this table:

Table 12.14 Boolean Values

True	False
ON	OFF
_ON_	_OFF_
1	0
TRUE	FALSE
YES	NO
YES	_NO_

Table 12.15 Table Attributes

Task	Attribute	Destinations
Influence the layout of the table		
Specify whether to try to place the same number of columns in each data panel if the entire table does not fit in one data panel	BALANCE	LISTING, printer family, and RTF
Specify whether to center each data panel independently if the entire table does not fit in one data panel	CENTER	LISTING, printer family, RTF
Specify whether to force a new page before printing the table	NEWPAGE	All except OUTPUT
Specify the number of sets of columns to place on a page	PANELS=	LISTING and printer family
Specify the number of blank characters to place between sets of columns when PANELS= is in effect	PANEL_SPACE=	LISTING
Specify the number of lines that must be available on the page in order to print the body of the table	REQUIRED_SPACE=	LISTING and printer family
Specify the number of lines to place between the previous output object and the current one	TOP_SPACE=	LISTING and printer family
Specify whether to split a table that is too wide to fit in the space that is provided or to wrap each row of the table	WRAP	LISTING and printer family
Specify whether to add a double space after the last line of a single row when the row is wrapped	WRAP_SPACE	LISTING and printer family
Influence the layout of rows and columns		
Specify the maximum number of blank characters to place between columns	COL_SPACE_MAX=	LISTING

Task	Attribute	Destinations
Specify the minimum number of blank characters to place between columns	COL_SPACE_MIN=	LISTING
Specify the name of the column whose value provides formatting information about the space before each row of the template	CONTROL=	All except OUTPUT
Specify whether to double space between the rows of the table	DOUBLE_SPACE	LISTING
Specify whether extra space is evenly divided among all columns of the table	EVEN	LISTING
Specify whether to split a long stacked column across page boundaries	SPLIT_STACK	LISTING
Influence the display of the values in header cells and data cells		
Specify whether to suppress blanking the value in a column that is marked with the BLANK_DUPS column attribute if the value changes in a previous column that is also marked with the BLANK_DUPS attribute	CLASSLEVELS=	LISTING and printer family
Specify which format to use if both a column template and a data component specify a format	DATA_FORMAT_OVERRIDE	All
Specify whether to justify the format fields within the columns or to justify the values within the columns without regard to the format fields	JUSTIFY	LISTING
Specify whether to order the columns by their order in the data component	ORDER_DATA	All except OUTPUT
Specify the source of the values for the format width and the decimal width if they are not specified	USE_FORMAT_DEFAULTS	All
Use the column name as the column header if neither the column template nor the data component specifies a header	USE_NAME	All
Influence the layout of headers and footers		
Specify the number of blank lines to place between the last row of data and the first row of output	FOOTER_SPACE=	LISTING
Specify the number of blank lines to place between the last row of headers and the first row of data	HEADER_SPACE=	LISTING
Specify whether to draw a continuous line above the first table footer or, if there is no table footer, below the last row of data on a page	OVERLINE	LISTING

Task	Attribute	Destinations
Specify whether to print table footers and any overlining of the table footers	PRINT_FOOTERS	All except OUTPUT
Specify whether to print table headers and any underlining of the table headers	PRINT_HEADERS	All except OUTPUT
Specify whether to draw a continuous line under the last table header or, if there is no table header, then above the last row of data on a page	UNDERLINE	LISTING
Influence the HTML output		
Specify whether to place the output object in a table of contents, if you create a table of contents	CONTENTS	HTML
Specify the label to use for the output object in the contents file, the Results window, and the trace record	CONTENTS_LABEL=	HTML, PDF, PRINTER, PS PDFMARK
Other table attributes		
Specify an alternate description for the table *	ALT=	MARKUP
Control whether BY lines are printed above each BY group	BYLINE=	All except OUTPUT
Define the characters to use as the line-drawing characters in the table	FORMCHAR=	LISTING
Specify a label for the table	LABEL=	All
Specify a long description for the table *	LONGDESC=	MARKUP
Specify the table that the current template inherits from	PARENT=	All
Specify the style element to use for the table and any changes to the attributes	STYLE=	Markup family, printer family, and RTF
Specify the special data set type of a SAS data set	TYPE=	OUTPUT

* SAS includes these accessibility and compatibility features that improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

ALT= "*text*"

specifies an alternate description of the table.

Requirement: The text must be enclosed with quotation marks.

Tip: The ALT= attribute is valid only in markup family destinations.

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

BALANCE <=ON | OFF | *variable*>

specifies whether to try to place the same number of columns in each data panel if the entire table does not fit in one data panel.

Default: OFF

Tip: The BALANCE attribute is valid only in the LISTING, printer family, and RTF

BYLINE <=ON | OFF | *variable*>
controls whether BY lines are printed above each BY group in a configuration file, SAS invocation, OPTIONS statement, or Systems Options window.

Category: PROC OPTIONS GROUP= LISTCONTROL

Default: OFF

Restriction: This attribute applies only if the table is not the first one on the page. If BY-group processing is in effect, a byline automatically precedes the first table on the page.

Tip: The BYLINE attribute is valid in all destinations except the OUTPUT destination.

CENTER <=ON | OFF | *variable*>
specifies whether to center each data panel independently if the entire table does not fit in the space that is provided.

Default: ON

Tip: The CENTER attribute is valid only in the LISTING, printer family, and RTF destinations.

CLASSLEVELS <=ON | OFF | *variable*>
specifies whether to suppress blanking the value in a column that is marked with the BLANK_DUPS column attribute if the value changes in a previous column that is also marked with the BLANK_DUPS attribute.

Default: OFF

Tip: The CLASSLEVELS attribute is valid for all destinations except the OUTPUT destination.

Featured in: Example 1 on page 551

COL_SPACE_MAX= *positive-integer* | *variable*
specifies the maximum number of blank characters to place between the columns.

Default: 4

Tip: The COL_SPACE_MAX= table attribute is valid only in the LISTING destination.

COL_SPACE_MIN= *positive-integer* | *variable*
specifies the minimum number of blank characters to place between the columns.

Default: 2

Tip: The COL_SPACE_MIN= attribute is valid only in the LISTING destination.

CONTENTS <=ON | OFF | *variable*>
specifies whether to place the output object in a table of contents, if you create a table of contents.

Default: ON

Tip: The CONTENTS attribute is valid in markup family and printer family destinations.

CONTENTS_LABEL= "*string*" | *variable*
specifies the label to use for the output object in the contents file, the Results window, and the trace record.

Default: If the SAS system option LABEL is in effect, the default label is the object's label. If LABEL is not in effect, the default label is the object's name.

Tip: The CONTENTS_LABEL= attribute is valid only in markup family and printer family destinations.

CONTROL=*column-name* | *variable*
specifies the name of the column whose values provide formatting information about the space before each row of the template. The value of CONTROL= should be the name of a column of type character with a length equal to 1.

Table 12.16 Values in the Control Column

Column Control Value	Result
A digit from 1-9	The specified number of blank lines precedes the current row.
A hyphen (-)	A row of underlining precedes the current row.
"b" or "B"	ODS tries to insert a panel break if the entire table does not fit in the space that is provided.

Default: None

Tip: The CONTROL= attribute is valid in all destinations except the OUTPUT destination.

DATA_FORMAT_OVERRIDE<=ON | OFF | *variable*>
specifies which format to use if both a column template and a data component specify a format.

ON
uses the format that the data component specifies.

OFF
use the format that the column template specifies.

Default: OFF

Tip: The DATA_FORMAT_OVERRIDE attribute is valid in all destinations.

DOUBLE_SPACE<=ON | OFF | *variable*>
specifies whether to double space between the rows of the table.

Default: OFF

Tip: The DOUBLE_SPACE attribute is valid only in the LISTING destination.

Featured in: Example 1 on page 756 and Example 3 on page 769

EVEN<=ON | OFF | *variable*>
specifies whether extra space is evenly divided among all columns of the table.

Default: OFF

Tip: The EVEN attribute is valid only in the LISTING destination.LISTING

FOOTER_SPACE=0 | 1 | 2 | *variable*
specifies the number of blank lines to place between the last row of data and the first row of the table footer.

Default: 1

Tip: The FOOTER_SPACE= attribute is valid only in the LISTING destination.

FORMCHAR= "*string*" | *variable*
defines the characters to use as the line-drawing characters in the table. Currently, ODS uses only the second of the 20 possible formatting characters. This formatting character is used for underlining and overlining. To change the second formatting

character, specify both the first and second formatting characters. For example, this option assigns the asterisk (*) to the first formatting character, the plus sign (+) to the second character, and does not alter the remaining characters:

```
formchar="*+"
```

Default: The SAS system option FORMCHAR= specifies the default formatting characters.

Tip: Use any character in formatting-characters, including hexadecimal characters. If you use hexadecimal characters, then put an x after the closing quote. For example, this option assigns the hexadecimal character 2D to the first formatting character, the hexadecimal character 7C to the second character, and does not alter the remaining characters:

```
formchar="2D7C"x
```

Tip: The FORMCHAR= attribute is valid only in the LISTING destination.

HEADER_SPACE=0 | 1 | 2 | *variable*
specifies the number of blank lines to place between the last row of headers and the first row of data. A row of underscores is a header.

Default: 1

Tip: The HEADER_SPACE= attribute is valid only in the LISTING destination.

JUSTIFY<=ON | OFF | *variable*>
specifies whether to justify the format fields within the columns or to justify the values within the columns without regard to the format fields.

Default: OFF

Interaction: JUSTIFY=ON can interfere with decimal alignment.

Interaction: If the column is numeric, then values are aligned to the right if you specify JUSTIFY=OFF and JUST=C.

Interaction: All of the destinations except for the LISTING destination justify the values in columns as if JUSTIFY=ON for JUST=R and JUST=L.

Tip: If you translate numeric data to character data, you might need to use JUSTIFY= to align the data.

Main discussion: “Values in Table Columns and How They Are Justified” on page 754

Tip: The JUSTIFY attribute is valid only in the LISTING destination.

LABEL= "*text*" | *variable*
specifies a label for the table.

Default: ODS uses the first of the following that it finds:

- a label that the table template provides
- a label that the data component provides
- the first spanning header in the table.

Tip: The LABEL= attribute is valid in all destinations.

LONGDESC= "*string*"
specifies the long description of the table.

Requirement: The text must be enclosed with quotation marks.

Tip: The LONGDESC= attribute is valid only in markup family destinations.

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility

standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

NEWPAGE<=ON | OFF | *variable*>
specifies whether to force a new page before printing the table.

Default: OFF

Restriction: If the table is the first item on the page, ODS ignores this attribute.

Tip: The NEWPAGE attribute is valid in all destinations except the OUTPUT destination.

ORDER_DATA<=ON | OFF | *variable*>
specifies whether to order the columns by their order in the data component.

Default: OFF
When ORDER_DATA=OFF, the default order for columns is the order that they are specified in the COLUMN statement. If you omit a COLUMN statement, the default order for columns is the order in which you define them in the template.

Interaction: ORDER_DATA is most useful for ordering generic columns.

Tip: The ORDER_DATA attribute is valid in all destinations except the OUTPUT destination. The OUTPUT destination always uses the order of the columns in the data component when it creates an output data set.

OVERLINE<=ON | OFF | *variable*>
specifies whether to draw a continuous line above the first table footer or, if there is no table footer, below the last row of data on a page. The second formatting character is used to draw the line.

Default: OFF

Main discussion: See the discussion of the FORMCHAR= attribute on page 647.

See also: UNDERLINE= on page 651 (for tables), UNDERLINE= on page 613 (for columns), and OVERLINE= on page 610 (for columns)

Tip: The OVERLINE attribute is valid only in the LISTING destination.

Featured in: Example 1 on page 756

PANELS=*positive-integer* | *variable*
specifies the number of sets of columns to place on a page. If the width of all the columns is less than half of the linesize, display the data in multiple sets of columns so that rows that would otherwise appear on multiple pages appear on the same page.

Tip: If the number of panels that is specified is larger than the number of panels that can fit on the page, the template creates as many panels as it can. Let the table template put data in the maximum number of panels that can fit on the page by specifying a large number of panels (for example, 99).

Tip: The PANELS= attribute is valid only in LISTING and printer family destinations.

PANEL_SPACE=*positive-integer* | *variable*
specifies the number of blank characters to place between sets of columns when PANELS= is in effect.

Default: 2

Tip: The PANEL_SPACE= attribute is valid only in the LISTING destination.

PARENT=*table-path*
specifies the table that the current template inherits from. A *table-path* consists of one or more names, separated by periods. Each name represents a directory in a template store. (A template store is a type of SAS file.) The current template inherits from the specified table in the first template store in the current path that you can read from.

When you specify a parent, all of the attributes and statements that are specified in the parent's template are used in the current template unless the current template overrides them.

Tip: The PARENT= attribute is valid in all destinations.

PRINT_FOOTERS<=ON | OFF | *variable*>
specifies whether to print table footers and any overlining of the table footers.

Default: ON

See also: OVERLINE=

Tip: The PRINT_FOOTERS attribute is valid in all destinations except the OUTPUT destination.

PRINT_HEADERS<=ON | OFF | *variable*>
specifies whether to print the table headers and any underlining of the table headers.

Default: ON

Interaction: When used in a table template, PRINT_HEADERS affects only headers for the table, not the headers for individual columns. (See the discussion of the PRINT_HEADERS column attribute on page 611.)

Tip: The PRINT_HEADERS attribute is valid in all destinations except the OUTPUT destination.

See also: UNDERLINE=

REQUIRED_SPACE=*positive-integer* | *variable*
specifies the number of lines that must be available on the page in order to print the body of the table (The body of the table is the part of the table that contains the data. It does not include headers and footers.)

Default: 3

Tip: The REQUIRED_SPACE= attribute is valid in LISTING and printer family destinations.

SPLIT_STACK<=ON | OFF | *variable*>
specifies whether to split a long stacked column across page boundaries.

Default: OFF

Tip: The SPLIT_STACK attribute is valid only in the LISTING destinations.

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element and any changes to its attributes to use for the table.

style-element-name
is the name of the style element to use to display the table. The style element must be part of a style that is registered with the Output Delivery System. SAS provides some styles. You can create customized styles with PROC TEMPLATE (see "DEFINE STYLE Statement" on page 490). By default, ODS produces different parts of ODS output with different elements. For example, by default, a table is produced with the style element Table. The styles that SAS provides do not provide another style element that you would be likely to want to use instead of Table. However, you might have a user-defined style element at your site that would be appropriate to specify.

The style element provides the basis for displaying the table. Additional style attributes that you provide can modify the display.

style-element-name is either the name of a style element or a variable whose value is a style element.

See also: "Viewing the Contents of a Style" on page 538

See also: "Working with Styles" on page 538

style-attribute-specification
describes the style attribute to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

See also: "Style Attributes and Their Values" on page 498

Default: Table

Requirement: Specify either a *style-attribute-specification* or a *style-element-name* with the STYLE= option.

Tip: You can use braces ({ and }) instead of square brackets ([and]).

Tip: If you use the STYLE= attribute inside a quoted string, then add a space before or after the carriage return to prevent errors. SAS does not interpret a carriage return as a space. You must explicitly specify spaces in quoted strings.

Tip: The STYLE= attribute is valid only in the markup family, printer family, and RTF destinations.

TOP_SPACE=*positive-integer* | *variable*
specifies the number of lines to place between the previous output object and the current one.

Default: 1

Tip: The TOP_SPACE= attribute is valid only in LISTING and printer family destinations.

TYPE=*string* | *variable*
specifies special type of SAS data set.

Restriction: PROC TEMPLATE does *not* verify the following:

- a SAS data set type that you specify is a valid data set type
- the structure of the data set that you create is appropriate for the type that you have assigned

Tip: Most SAS data sets have no special type. However, certain SAS procedures, like the CORR procedure, can create a number of special SAS data sets. In addition, SAS/STAT software and SAS/EIS software support special data set types.

Tip: The TYPE= attribute is valid only in the OUTPUT destination.

UNDERLINE<=ON | OFF | *variable*>
specifies whether to draw a continuous line under the last table header (or, if there is no table header, then above the first row of data on a page). The second formatting character is used to draw the line.

Default: OFF

Main discussion: See the discussion of the FORMCHAR= attribute on page 647

See also: OVERLINE= (for tables) on page 649 , UNDERLINE (for columns) on page 613, and OVERLINE (for columns) on page 610

Tip: The UNDERLINE attribute is valid only in the LISTING destination.

Featured in: Example 1 on page 756 and Example 3 on page 769

USE_FORMAT_DEFAULTS<=ON | OFF | *variable*>
specifies the source of the values for the format width and the decimal width if they are not specified.

ON
uses the default values, if any, that are associated with the format name.

OFF
uses the PROC TEMPLATE defaults.

Default: OFF

Tip: The USE_FORMAT_DEFAULTS attribute is valid in all destinations except the OUTPUT destination.

USE_NAME<=ON | OFF | *variable*>
uses the column name as the column header if neither the column template nor the data component specifies a header.

Default: OFF

Tip: Use this attribute when column names are derived from a data set and the columns are generic.

Tip: The USE_NAME attribute is valid in all destinations except the OUTPUT destination.

WRAP<=ON | OFF | *variable*>
specifies whether to split a wide table into multiple data panels, or to wrap each row of the table so that an entire row is printed before the next row starts.

Default: OFF

Interaction: When ODS wraps the rows of a table, it does not place multiple values in any column that contains an ID column.

See also: WRAP_SPACE= and ID= on page 608

Tip: The WRAP attribute is valid only in LISTING and printer family destinations.

WRAP_SPACE<=ON | OFF | *variable*>
specifies whether to double space after the last line of a single row of the table when the row is wrapped onto more than one line.

Default: OFF

See also: WRAP=

Tip: The WRAP_SPACE attribute is valid only in the LISTING, printer family, and RTF destinations.

CELLSTYLE–AS Statement

Sets the style element of the cells in the table according to the values of the variables. Use this statement to set the presentation characteristics (such as foreground color and font face) of individual cells.

Featured in: Example 4 on page 777

CELLSTYLE *expression-1* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>
<…, *expression-n* **AS** <*style-element-name*><[*style-attribute-specification(s)*]>>;

Required Arguments

expression
is an expression that is evaluated for each table cell that contains a variable.

If *expression* resolves to TRUE (a non-zero value), the style element that is specified is used for the current cell. If *expression* is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

expression has this form:

expression-1 <comparison-operator expression-n>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

SAS function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

Built-in variable
is a special kind of WHERE expression operand that helps you find common values in table templates. Built-in variables are one or more of the following:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data column name.

DATATYPE
is the data type of the column variable. The data type is either numeric ("num") or character ("char").

Example: The following CELLSTYLE AS statement specifies that numeric column variables have a red font color and character column variables have a blue font color:

```
cellstyle   _datatype_ = "num" as {color=red},
             _datatype_ = "char" as {color=blue};
```

LABEL
is a column label.

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style element name.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 12.17 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Featured in: Example 5 on page 782

Tip: Using an expression of 1 as the last expression in the CELLSTYLE AS statement sets the style element for any cells that did not meet an earlier condition.

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

style-attribute-specification
describes a style attribute to set. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

For information on the style attributes that you can set in a table template, see "Style Attributes and Their Values" on page 498.

Options

style-element-name
is the name of a style element that is part of a style that is registered with the Output Delivery System. SAS provides some styles. You can create customized styles and style elements with PROC TEMPLATE. (See "DEFINE STYLE Statement" on page 490.)

The style elements that you would be most likely to use with the CELLSTYLE AS statement are

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataEmphasisFixed
- DataStrong

- DataStrongFixed

The style element provides the basis for displaying the cell. Additional style attributes modify the display.

COLUMN Statement

Declares a symbol as a column in the table and specifies the order of the columns.

Featured in: Example 3 on page 769

COLUMN *column(s)*;

Required Arguments

column
is one or more columns. If the column is defined outside the current table template, reference it by its path in the template store. Columns in the template are laid out from left to right in the same order that they are specified in the COLUMN statement.

Default: If you omit a COLUMN statement, ODS makes a column for each column template (DEFINE COLUMN statement), and places the columns in the same order that the column templates have in the table template.

If you use a COLUMN statement but omit a DEFINE COLUMN statement for any of the columns, ODS uses a default column template that is based on the type of data in the column.

Interaction: If you specify the column attribute PRINT=OFF, then the value of a column is turned off if the column is part of a stacked column. If all columns in a stacked column have PRINT=OFF set, then the entire column is removed from the table.

Tip: Use a list of variable names, such as DAY1–DAY10, to specify multiple variables.

Main discussion: *Stacking Values for Two or More Variables*

To stack values for two or more variables in the same column, put parentheses around the stacked variables. In such a case, the column header for the first column inside the parentheses becomes the header for the column that contains all the variables inside parentheses. For example, this COLUMN statement produces a template with the following characteristics:

- The value of NAME is in the first column by itself.
- The values of CITY and STATE appear in the second column with CITY above STATE. The header for this column is the header that is associated with CITY.
- The values HOMEPHONE and WORKPHONE appear in the third column with HOMEPHONE above WORKPHONE. The header for this column is the header that is associated with HOMEPHONE.

```
column name (city state) (homephone workphone);
```

Use the asterisk (*) in the COLUMN statement to change the layout of stacking variables. An asterisk between groups of variables in parentheses stacks the first item in the first set of parentheses above the first item in the next set of parentheses, and so on until the last group of parentheses is reached. Then, the second item in the first group is stacked above the second item in the second group, and so on. For example, this COLUMN statement produces a report with the following characteristics:

- The value of NAME is in the first column by itself.
- The values of CITY and HOMEPHONE appear in the second column with CITY above HOMEPHONE. The header for this column is the header that is associated with CITY.
- The values STATE and WORKPHONE appear in the third column with STATE above WORKPHONE. The header for this column is the header that is associated with STATE.

```
column name (city state) * (homephone workphone);
```

DEFINE Statement

Creates a template inside a table template.

Main discussion: "DEFINE COLUMN Statement" on page 599, "DEFINE FOOTER Statement" on page 625, and "DEFINE HEADER Statement" on page 626

DEFINE *template-type template-name*</ *option(s)*>;
statements-and-attributes;
END;

Required Arguments

template-type
specifies the type of template to create, where *template-type* is one of the following:

COLUMN

FOOTER

HEADER

The *template-type* determines what other statements and what attributes can go in the template. For details, see the documentation for the corresponding DEFINE statement.

template-name
specifies the name of the new object.

Restriction: *template-name* must be a single-level name.

Note: To reference the template that you are creating from another template, create it outside the table template. △

Options

NOLIST
preserves the *template-type* when inheriting it from another table template.

Tip: If you specify an existing *template-name* without using the NOLIST option, then the template is overwritten.

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Scope: You can use the DYNAMIC statement in the template of a table, column, header, or footer. A dynamic variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 1 on page 551 and Example 2 on page 557

DYNAMIC *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a variable that the data component supplies. ODS resolves the value of the variable when it binds the template and the data component.

Tip: Dynamic variables are most useful to the authors of SAS procedures and to DATA step programmers.

Options

text
is text that is placed in the template to explain the dynamic variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

HEADER Statement

Declares a symbol as a header in the table and specifies the order of the headers.

HEADER *header-specification(s)*;

Required Arguments

header-specification
is one or more headers. If the header is defined outside the current table template, reference it by its path in the template store. Headers in the template are laid out

from top to bottom in the same order that they are specified in the HEADER statement. Each *header-specification* is one of the following:

"*string*"
specifies the text to use for the header. If you specify a string, you do not need to use a DEFINE HEADER statement. However, you cannot specify any header attributes except for a split character. If the SPLIT= header attribute is not in effect and if the first character of the header that you specify is neither a blank character nor an alphanumeric character, PROC TEMPLATE treats it as the split character.

See also: SPLIT= on page 634

header-path
is the path of the header template to use. A header-path consists of one or more names, separated by periods. Each name represents a directory in a template store. (A template store is a type of SAS file.)

LABEL
uses the label of the output object as the header. Each SAS procedure specifies a label for each output object that it creates. The DATA step uses the value of the OBJECTLABEL= option as the label of the output object. If OBJECTLABEL= is not specified, it uses the text of the first TITLE statement as the label.

Default: If you omit a HEADER statement, then ODS makes a header for each header template (DEFINE HEADER statement), and places the headers in the same order that the header templates have in the table template.

Featured in: Example 3 on page 769

FOOTER Statement

Declares a symbol as a footer in the table and specifies the order of the footers.

FOOTER *footer-specification(s)*;

Required Arguments

footer-specification
is one or more footers. If the footer is defined outside the current table template, reference it by its path in the template store. Footers in the template are laid out from top to bottom in the same order that they are specified in the FOOTER statement. Each *footer-specification* is one of the following:

"*string*"
specifies the text to use for the footer. If you specify a string, you do not need to specify a DEFINE FOOTER statement. However, you cannot specify any footer attributes except for a split character. If the SPLIT= attribute is not in effect and if the first character of the footer that you specify is neither a blank character nor an alphanumeric character, PROC TEMPLATE treats it as the split character.

See also: SPLIT= on page 634

footer-path

is the path of the footer template to use. A footer-path consists of one or more names, separated by periods. Each name represents a directory in a template store, which is s a type of SAS file.

LABEL
uses the label of the output object as the footer. Each SAS procedure specifies a label for each output object that it creates. The DATA step uses the value of the OBJECTLABEL= option as the label of the output object. If OBJECTLABEL= is not specified, it uses the text of the first TITLE statement as the label.

Default: If you omit a FOOTER statement, ODS makes a footer for each footer template (DEFINE FOOTER statement), and places the footers in the same order that the footer templates have in the table template.

MVAR Statement

Defines a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.

Scope: You can use the MVAR statement in the template of a table, column, header, or footer. A macro variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 3 on page 769 and Example 1 on page 551

MVAR *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a macro variable to reference in the template. ODS will use the value of the macro variable as a string. ODS does not resolve the value of the macro variable until it binds the template and the data component.

Tip: Declare macro variables this way in a template. For example, to use the automatic macro variable SYSDATE9 in a template, declare it in an MVAR statement and reference it as SYSDATE9, without an ampersand, in the PROC TEMPLATE step. If you use the ampersand, the macro variable resolves when the template is compiled instead of when ODS binds the template to the data component.

Options

text
is text that is placed in the template to explain the macro variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

NMVAR Statement

Defines a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the template and the data component to produce an output object.

Scope: You can use the NMVAR statement in the template of a table, column, header, or footer. A macro variable that is defined in a template is available to that template and to all the templates that it contains.

Featured in: Example 4 on page 777

NMVAR *variable-1 <'text-1'> <… variable-n <'text-n'>>*;

Required Arguments

variable
names a macro variable to reference in the template. ODS will convert the variable's value to a number (stored as a double) before using it. ODS does not resolve the macro variable until it binds the template and the data component.

Tip: Declare macro variables this way in a template. For example, to use a macro variable as a number, declare it in an NMVAR statement and reference it without an ampersand. If you use the ampersand, the macro variable resolves when the template is compiled instead of when ODS binds the template to the data component.

Options

text
is text that is placed in the template to explain the macro variable's use. Text of this type becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES Statement

Provides information about the table, header, column, or footer.

Tip: The NOTES statement becomes part of the compiled template, which you can view with the SOURCE statement, whereas SAS comments do not.

Featured in: Example 4 on page 777

NOTES *'text'*;

Required Arguments

text
provides information about the table.

TRANSLATE INTO Statement

Translates the specified numeric values to other values.

Restriction: The TRANSLATE INTO statement in a table template applies only to numeric variables. To translate the values of a character variable, use TRANSLATE INTO in the template of that column. (See "DEFINE COLUMN Statement" on page 599).

Featured in: Example 4 on page 777

TRANSLATE *expression-1* **INTO** *expression-2* <…, *expression-n* **INTO** *expression-m*>;

Required Arguments

expression-1
is an expression that is evaluated for each table cell that contains a numeric variable.

If *expression-1* resolves to TRUE (a non-zero value), the translation that is specified is used for the current cell. If *expression-1* is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

expression has this form:

expression-1 <*comparison-operator expression-n*>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

SAS function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

Built-in variable
is a special kind of WHERE expression operand that helps you find common values in table templates. Built-in variables are one or more of the following:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data column name.

DATATYPE

is the data type of the column variable. The data type is either numeric ("num") or character ("char").

LABEL
is a column label.

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style element name.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 12.18 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Restriction: You cannot reference the values of other columns in *expression-1*.

Tip: Using an expression of 1 as the last expression in the TRANSLATE–INTO statement specifies a translation for any cells that did not meet an earlier condition.

Featured in: Example 5 on page 782

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

expression-2
is an expression that specifies the value to use in the cell in place of the variable's actual value.

expression has this form:

expression-1 <comparison-operator expression-n>

expression
is an arithmetic or logical expression that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

constant
is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

SAS function
specifies a SAS function. For information on SAS functions, see *SAS Language Reference: Dictionary*.

Built-in variable
a special kind of WHERE expression operand that helps you find common values in table templates. Built-in variables are one or more of the following:

COLUMN
is a column number. Column numbering begins with 1.

Alias: _COL_

Featured in: Example 5 on page 782

DATANAME
is a data column name.

DATATYPE
is the data type of the column variable. The data type is either numeric ("num") or character ("char").

LABEL
is a column label

Featured in: Example 5 on page 782

ROW
is a row number. Row numbering begins with 1.

Featured in: Example 5 on page 782

STYLE
is a style element name.

Featured in: Example 6 on page 788

VAL
is the data value of a cell.

Tip: Use _VAL_ to represent the value of the current column.

Featured in: Example 6 on page 788

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 12.19 Comparison Operators

Symbol	Mnemonic Equivalent	Definition
=	EQ	Equal to
^= or ~= or ¬= or <>	NE	Not equal to
>	GT	Greater than

Symbol	Mnemonic Equivalent	Definition
<	LT	Less than
>=	GE	Greater than or equal to
<=	LE	Less than or equal to
	IN	Equal to one from a list of values

Restriction: *expression-2* must resolve to a character value, not a numeric value.

Tip: When you translate a numeric value to a character value, the table template does not try to apply the numeric format that is associated with the column. Instead, it simply writes the character value into the formatted field, starting at the left. To right-justify the value, use the JUSTIFY=ON attribute.

Featured in: Example 5 on page 782

See also: JUSTIFY= on page 648

See also: You can use any expression that can be used in the WHERE= data set option. For information on expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

END Statement

Ends the table template, header template, column template, or footer template.

END;

ODS Output Object Table Names

Some SAS procedures assign names to the tables that they create. When using ODS, you can select tables and create output data sets by referencing these names. The following tables list the output object table names that Base SAS, SAS/STAT, and SAS/ETS procedures produce:

- "ODS Table Names and the Base SAS Procedures That Produce Them" on page 664
- "ODS Table Names and the SAS/STAT Procedures That Produce Them" on page 673
- "ODS Table Names and the SAS/ETS Procedures That Produce Them" on page 731

ODS Table Names and the Base SAS Procedures That Produce Them

This table lists the output object table names which Base SAS procedures produce. The table provides the name of each table, a description of what the table contains, and the option, if any, that creates the output object table. For more information about Base SAS procedures, see *Base SAS Procedures Guide*.

Table 12.20 ODS Table Names Produced by the CALENDAR Procedure

Table Name	Description
Calendar	Calendar

Table 12.21 ODS Table Names Produced by the CATALOG Procedure

Table Name	Description
Catalog_Random	Table generated when the catalog is in a random-access data library
Catalog_Sequential	Table generated when the catalog is in a sequential-access data library

Table 12.22 ODS Table Names Produced by the CHART Procedure

Table Name	Description
Block	Block chart
Hbar	Horizontal bar chart
Pie	Pie chart
Star	Star chart
Vbar	Vertical bar chart

Table 12.23 ODS Table Names Produced by the COMPARE Procedure

Table Name	Description	Option
CompareDatasets	Information about the data set or data sets	Omit NOSUMMARY or NOVALUE options
CompareDetails (Comparison results for observations)	List of observations that the base data set and the compare data set do not have in common	PRINTALL

Table Name	Description	Option
CompareDifferences	Report of variable value differences	Omit NOVALUES option
CompareSummary	Summary report of observations, values, and variables of unequal values	
CompareVariables	List of differences in variable types or attributes between the base data set and the compare data set	Omit NOSUMMARY option or unless the variables are identical
	ODS Tables Created by the ID Statement	
CompareDetails	List of notes and warnings concerning duplicate ID variable values, if duplicate ID variable values exist in either the data set	

Table 12.24 ODS Table Names Produced by the CORR Procedure

Table Name	Description	Option
Cov	Covariance table row/column variance DF (missing values)	COV
CronbachAlpha	Coefficient alpha	ALPHA
CronbachAlphaDel	Coefficient alpha with deleted variables	ALPHA
Csscp	Corrected sums of squares and crossproducts Row/column variable corrected sums of squares (missing values)	CSSCP
HoeffdingCorr	Hoeffding's D statistics p-value (NOPROB is not specified) Number of observations (missing values)	HOEFFDING
KendallCorr	Kendall tau-b coefficients p-value (NOPROB is not specified) Number of observations (missing values)	Pearson or omit NOCORR option

Table Name	Description	Option
SimpleStats	Simple descriptive statistics	Omit NOSIMPLE option
SpearmanCorr	Spearman descriptive statistics	SPEARMAN
Sscp	Sums of squares and crossproducts	SSCP
	Row/column variable sums of squares (missing values)	
VarInformation	Variable information	
ODS Tables Created by the PARTIAL Statement		
PartialCsscp	Partial corrected sums of squares and crossproducts	CSSCP
PartialCov	Partial covariances	COV
PartialKendallCorr	Partial Kendall tau-b coefficients	KENDALL
PartialPearsonCorr	Partial Kendall tau-b coefficients	
	p-values (NOPROB option is not specified)	
PartialSpearmanCorr	Partial Spearman correlations	SPEARMAN
	p-values (NOPROB option is not specified)	

Table 12.25 ODS Table Names Produced by the DATASETS and CONTENTS Procedures

Table Name	Description	Option
Directory	General library information	Omit NOLIST option
Members	Library member information	Omit NOLIST option

Table 12.26 ODS Table Names Produced by the CONTENTS Procedure or the DATASETS Procedure with the CONTENTS Statement

Table Name	Description	Option
Attributes	Data set attributes	Omit SHORT option
Directory	General library information	DATA=<*libref*.>_ALL_ or the DIRECTORY option*

Table Name	Description	Option
EngineHost	Engine and operating environment information	Omit SHORT option
IntegrityConstraints	List of integrity constraints	Omit SHORT option and data has integrity constraints
IntegrityConstraintsShort	Concise listing of integrity constraints	SHORT option specified and data has integrity constraints
Indexes	List of indexes	Omit SHORT option and data set is indexed
IndexesShort	Concise list of indexes	SHORT option specified and data set is indexed
Members	Library member information	DATA=<*libref*.>_ALL_ or the DIRECTORY option*
Position	List of variables by logical position in the data set	Omit SHORT option and specify the VARNUM option
PositionShort	Concise list of variables by logical position in the data set	SHORT and VARNUM options
Sortedby	Sort information	Omit SHORT option and data set is sorted
SortedbyShort	Concise sort information	SHORT option and data set is sorted
Variables	List of variables in alphabetical order	Omit SHORT option
VariablesShort	Concise listing of variables in alphabetical order	SHORT

* For PROC DATASETS, if both the NOLIST option and either the DIRECTORY option or DATA=<*libref*.>_ALL_ are specified, then the NOLIST option is ignored.

Table 12.27 ODS Table Names Produced by the FREQ Procedure

Table Name	Description	Option
BinomialCLs	Binomial confidence limits	BINOMIAL(AC | J | W)
BinomialEquiv	Binomial equivalence analysis	BINOMIAL(EQUIV)
BinomialEquivLimits	Binomial equivalence limits	BINOMIAL(EQUIV)
BinomialEquivTest	Binomial equivalence test	BINOMIAL(EQUIV)
BinomialNoninf	Binomial noninferiority test	BINOMIAL(NONINF)
BinomialPropTest	Binomial proportion test	BINOMIAL (one-way tables)
BinomialProp	Binomial proportion	BINOMIAL (one-way tables)
BinomialSup	Binomial superiority test	BINOMIAL(SUP)
BreslowDayTest	Breslow-day test	CMH (hx2x2 tables)

Table Name	Description	Option
CMH	Cochran-Mantel-Haenszel statistics	CMH
ChiSq	Chi-Square tests and measures	CHISQ
CochransQ	Cochran's Q	AGREE (hx2x2 tables)
ColScores	Column scores	SCOROUT
CommonOddsRatioCL	Exact confidence limits for the common odds ratio	COMOR (hx2x2 tables)
CommonOddsRatioTest	Common odds ratio exact test	(hx2x2 tables)
CommonRelRisks	Common relative risks	CMH (hx2x2 tables)
Crosslist	Cross lists	CROSSLIST (n-way table request, n>1)
CrossTabFreqs	Crosstabulation table	(n-way table request, n>1)
EqualKappaTest	Test for equal simple kappas	AGREE (hx2x2 tables)
EqualKappaTests	Test for equal kappas	AGREE (hxrxr tables, r>2)
EqualOddsRatios	Tests for equal odds ratios	EQOR (hx2x2 tables)
FishersExact	Fisher's exact test	FISHER or EXACT or CHISQ (2x2 tables)
FishersExactMC	Monte Carlo estimates for Fisher's exact test	FISHER / MC
Gamma	Gamma	GAMMA
GammaTest	Gamma Test	GAMMA
JTTest	Jonckheere-Terpstra test	JT
JTTestMC	Monte Carlo estimates for Jonckheere-Terpstra exact test	JT / MC
KappaStatistics	Kappa statistics	AGREE (rxr tables, r>2, and no TEST or EXACT requests for kappas)
KappaWeight	Kappa weights	AGREE and PRINTKWT
List	List frequencies	LIST
LRChiSq	Likelihood-ratio chi-square exact test	LRCHI
LRChiSqMC	Monte Carlo exact test for likelihood-ratio chi-square	LRCHI / MC
McNemarsTest	McNemar's test	AGREE (2x2 tables)
Measures	Measures of association	MEASURES
MHChiSq	Mantel-Haenszel chi-square exact test	MHCHI
MHChiSqMC	Monte Carlo exact test for Mantel-Haenszel chi-square	MHCHI / MC

Table Name	Description	Option
NLevels	Number of variable levels	NLEVELS
OddsRatioCL	Exact confidence limits for the odds ratio	OR (2x2 tables)
OneWayChiSq	One-way chi-square test	CHISQ (one-way tables)
OneWayChiSqMC	Monte Carlo exact test for one-way chi-square	CHISQ / MC (one-way tables)
OneWayFreqs	One-way frequencies	(One-way table request)
OverallKappa	Overall simple kappa coefficient	AGREE (hx2x2 tables)
Overallkappas	Overall kappa coefficients	AGREE (hxrxr tables, r>2)
PdiffEquiv	Equivalence analysis for the proportion difference	RISKDIFF(EQUIV) (2x2 tables)
PdiffEquivLimits	Equivalence limits for the proportion difference	RISKDIFF(EQUIV) (2x2 tables)
PdiffEquivTest	Equivalence test for the proportion difference	RISKDIFF(EQUIV) (2x2 tables)
PdiffNoninf	Noninferiority test for the proportion difference	RISKDIFF(NONINF) (2x2 tables)
PdiffSup	Superiority test for the proportion difference	RISKDIFF(SUP) (2x2 tables)
PdiffTest	Proportion difference test	RISKDIFF(EQUAL) (2x2 tables)
PearsonChiSq	Pearson chi-square exact test	PCHI
PearsonChiSqMC	Monte Carlo exact test for Pearson chi-square exact test	PCHI / MC
PearsonCorr	Pearson correlation	PCORR
PearsonCorrMC	Monte Carlo exact test for Pearson correlation	PCORR / MC
PearsonCorrTest	Pearson correlation test	PCORR
RelativeRisks	Relative risk estimates	RELRISK or MEASURES (2x2 tables)
RiskDiffCol1	Column 1 risk estimates	RISKDIFF (2x2 tables)
RiskDiffCol2	Column 2 risk estimates	RISKDIFF (2x2 tables)
RowScores	Row scores	SCOROUT
SimpleKappa	Simple kappa coefficient	KAPPA
SimpleKappaMC	Monte Carlo exact test Simple kappa coefficient	KAPPA / MC
SimpleKappaTest	Simple kappa tests	KAPPA ,
SomersDCR	Somers' *D(C | R)*	SMDCR
SomersDCRTest	Somers' *D(C | R)* test	SMDCR

Table Name	Description	Option
SomersDRC	Somers' *D(R | C)*	SMDRC
SomersDRCTest	Somers' *D(R | C) test*	SMDRC
SpearmanCorr	Spearman correlation	SCORR
SpearmanCorrMC	Monte Carlo exact test Spearman correlation	SCORR / MC
SpearmanCorrTest	Spearman correlation test	SCORR
SymmetryTest	Test of symmetry	AGREE
TauB	Kendall's tau-*b*	KENTB
TauBTest	Kendall's tau-*b* test	KENTB
TauC	Stuart's tau-*c*	STUTC
TauCTest	Stuart's tau-*c* test	STUTC
TrendTest	Cochran-Armitage test for trend	TREND
TrendTestMC	Monte Carlo exact test for trend	TREND / MC
WeightKappa	Weighted kappa coefficient	AGREE (rxr tables, r>2)
WeightedKappaMC	Monte Carlo exact test for weighted kappa	WTKAP / MC
WeightedKappaTest	Weighted kappa test	WTKAP

Table 12.28 ODS Table Names Produced by the MEANS and SUMMARY Procedures

Table Name	Description
Summary	Summary of descriptive statistics for variables across all observations and within groups of observations

Table 12.29 ODS Table Names Produced by the PLOT Procedure

Table Name	Description	Option
Plot	Single plot graph	
Overlaid	two or more plots on a single set of axes	OVERLAY

Table 12.30 ODS Table Names Produced by the REPORT Procedure

Table Name	Description
Report	Detail report, summary report, or combination of both detail and summary information report

Table 12.31 ODS Table Names Produced by the SQL Procedure

Table Name	Description
SQL_Results	SAS data file or a SAS data view

Table 12.32 ODS Table Names Produced by the TABULATE Procedure

Table Name	Description
Table	Descriptive statistics in tabular format that use some or all of the variables in a data set

Table 12.33 ODS Table Names Produced by the TIMEPLOT Procedure

Table Name	Description	Option
Plot	Single plot graph	Omit the OVERLAY option
OverlaidPlot	Two or more plots on a single set of axes	OVERLAY

Table 12.34 ODS Table Names Produced by the UNIVARIATE Procedure

Table Name	Description	Option
	ODS Tables Created by the PROC UNIVARIATE Statement	
BasicIntervals	Confidence intervals for mean, standard deviation, variance	CIBASIC
BasicMeasures	Measures of location and variability	default
ExtremeObs	Extreme observations	default
ExtremeValues	Extreme values	NEXTRAVAL=
Frequencies	Frequencies	FREQ
LocationCounts	Counts used for sign test and signed rank test	LOCCOUNT
Missing Values	Missing values	default, if missing values exist
Modes	Modes	MODES
Moments	Sample moments	default
Plots	Line printer plots	PLOTS
Quantiles	Quantiles	default
RobustScale	Robust measures of scale	ROBUSTSCALE
SSPlots	Line printer side-by-side box plot	PLOTS with BY statement
TestsForLocation	Tests for location	default
TestsForNormality	Tests for normality	NORMALTEST
TrimmedMeans	Trimmed means	TRIMMED=
WinsorizedMeans	Winsorized means	WINSORIZED=
	ODS Tables Created by the HISTOGRAM Statement	
Bins	histogram bins	MIDPERCENTS secondary option
FitQuantiles	quantiles of fitted distribution	any distribution option
GoodnessOfFit	goodness-of-fit tests for fitted distribution	any distribution option
HistogramBins	histogram bins	MIDPERCENTS option
ParameterEstimates	parameter estimates for fitted distribution	any distribution option

ODS Table Names and the SAS/STAT Procedures That Produce Them

This table lists the output object table names which SAS/STAT procedures produce. You must license SAS/STAT software in order to produce these output objects. The table provides the name of each table, a description of what the table contains, and the

option, if any, that creates the output object table. For information about SAS/STAT procedures, see *SAS/STAT User's Guide*.

Table 12.35 ODS Table Names Produced by the ACECLUS Procedure

Table Name	Description	Option
	ODS Tables Created by the PROC Statement	
ConvergenceStatus	Convergence status	
DataOptionInfo	Data and option information	
Eigenvalues	Eigenvalues of Inv(ACE)*(COV-ACE)	
Eigenvectors	Eigenvectors (raw canonical coefficients)	
InitWithin	Initial within-cluster covariances estimate	INITIAL=INPUT
IterHistory	Iteration history	
SimpleStatistics	Simple statistics	
StdCanCoef	Standardized canonical coefficients	
Threshold	Threshold value	PROPORTION=
TotSampleCov	Total sample covariances	
Within	Approximate covariance estimate within clusters	

Table 12.36 ODS Table Names Produced by the ANOVA Procedure

Table Name	Description	Option
DependentInfo	Simultaneously analyzed dependent variables	Default when there are multiple dependent variables with different patterns of missing values
FitStatistics	R-Square, C.V., root MSE, and dependent mean	
ModelANOVA	ANOVA for model terms	
NObs	Number of observations	
OverallANOVA	Overall ANOVA	

Table Name	Description	Option
	ODS Tables Created by the CLASS Statement	
ClassLevels	Classification variable levels	
	ODS Tables Created by the MANOVA Statement	
MANOVATransform	Multivariate transformation matrix	M=
MultStat	Multivariate tests	
Tests	Summary ANOVA for specified MANOVA H= effects	H=SUMMARY
	ODS Tables Created by the MANOVA or REPEATED Statements	
CanAnalysis	Canonical analysis	CANONICAL
CanCoef	Canonical coefficients	CANONICAL
CanStructure	Canonical structure	CANONICAL
CharStruct	Characteristic roots and vectors	MANOVA (not CANONICAL); REPEATED PRINTRV
ErrorSSCP	Error SSCP matrix	PRINTE
HypothesisSSCP	Hypothesis SSCP matrix	PRINTE; MANOVA M=
PartialCorr	Partial correlation matrix	PRINTE; REPEATED (CONTRAST, HELMERT, MEAN, POLYNOMIAL, or PROFILE)
	ODS Tables Created by the MEANS Statement	
Bartlett	Bartlett's homogeneity of variance test	HOVTEST=BARTLETT
CLDiffs	Multiple comparisons of pairwise differences	CLDIFF or DUNNETT or (Unequal cells and not LINES)
CLDiffsInfo	Information for multiple comparisons of pairwise differences	CLDIFF or DUNNETT or (Unequal cells and not LINES)
CLMeans	Multiple comparisons of means with confidence/comparison	CLM with (BON, GABRIEL, SCHEFFE, SIDAL. SMM, T, or LSD)
CLMeansInfo	Information for multiple comparisons of means with confidence/comparison interval	CLM
HOVFTest	Homogeneity of variance ANOVA	HOVTEST

Table Name	Description	Option
MCLines	Multiple comparisons LINES output	LINES, ((DUNCAN or WALLER or SNK or REGWQ) and not (CLDIFF or CLM)), or (equal cells and not CLDIFF)
MCLinesInfo	Information for multiple comparison LINES output	LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (equal cells and not CLDIFF
MCLinesRange	Ranges for multiple range MC tests	LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (equal cells and not CLDIFF)
Means	Group means	
Welch	Welch's ANOVA	WELCH
ODS Tables Created by the REPEATED Statement		
Epsilons	Greenhouse-Geisser and Huynh-Feldt epsilons	
RepTransform	Repeated transformation matrix	CONTRAST, HELMERT, MEAN, POLYNOMIAL, or PROFILE
RepeatedLevelInfo	Correspondence between dependents and repeated measures levels	
Sphericity	Sphericity tests	PRINTE
ODS Tables Created by the TEST Statement		
AltErrTests	ANOVA tests with error other than MSE	E=

Table 12.37 ODS Table Names Produced by the CALIS Procedure

Table Name	Description	Option
ODS Tables Created by the COSAN, FACTOR, LINEQS, and RAM Models		
AddParms	Additional parameters in the PARAMETERS statement	PINITIAL or default
AsymStdRes	Asymptotically standardized residual matrix	RESIDUAL= or PRINT

Table Name	Description	Option
AveAsymStdRes	Average absolute asymptotically standardized residuals	RESIDUAL= or PRINT
AveNormRes	Average absolute normalized residuals	RESIDUAL= or PRINT
AveRawRes	Average absolute raw residuals	RESIDUAL= or PRINT
AveVarStdRes	Average absolute variance standardized residuals	RESIDUAL= or PRINT
ContKurtosis	Contributions to kurtosis	KURTOSIS or PRINT
ConvergenceStatus	Convergence status	PSHORT
CorrParm	Correlations among parameter estimates	PCOVES and default
CovMat	Assorted cov matrices	PCOVES and default
DependParms	Dependent parameters (if specified by program statements)	PRIVEC and default
DistAsymStdRes	Distribution of asymptotically standardized residuals	RESIDUAL= or PRINT
DistNormRes	Distribution of normalized residuals	RESIDUAL= or PRINT
DistVarStdRes	Distribution of variance standardized residuals	RESIDUAL= or PRINT
Estimates	Vector of estimates	PRIVEC
Fit	Fit statistics	PSUMMARY
GenModInfo	General modeling information	PSIMPLE or default
Gradient	First partial derivatives (Gradient)	PRIVEC and default
InCorr	Input correlation matrix	PCORR or PALL
InCorrDet	Determinant of the input correlation matrix	PCORR or PALL
InCov	Input covariance matrix	PCORR or PALL
InCovDet	Determinant of the input covariance matrix	PCORR or PALL
Information	Information matrix	PCOVES and default
InitEstimates	Initial vector of parameter estimates	PINITIAL or default
InSymmetric	Input symmetric matrix (SYMATRIX data type)	PCORR or PALL
IterHist	Iteration history	PSHORT
IterStart	Iteration start	PSHORT

Table Name	Description	Option
IterStop	Iteration stop	PSHORT
Jacobian	Jacobi column pattern	PJACPAT
Kurtosis	Kurtosis, with raw data input	KURTOSIS or PRINT
LagrangeBoundary	Lagrange, releasing active boundary constraints	MODIFICATION or PALL
LagrangeEquality	Lagrange, releasing equality constraints	MODIFICATION or PALL
ModelStatement	Model summary	PSHORT
ModIndices	Lagrange multiplier and Wald test statistics	MODIFICATION or PALL
NormRes	Normalized residual matrix	RESIDUAL= or PRINT
PredetElements	Predetermined elements	PREDET or PALL
PredModel	Predicted model matrix	PCORR or PALL
PredModelDet	Predicted model determinant	PCORR or PALL
ProblemDescription	Problem Description	PSHORT
RankAsymStdRes	Ranking of the largest asymptotically standardized residuals	RESIDUAL= or PRINT
RankLagrange	Ranking of the largest Lagrange indices	RESIDUAL= or PRINT
RankNormRes	Ranking of the largest normalized residuals	RESIDUAL= or PRINT
RankRawRes	Ranking of the largest raw residuals	RESIDUAL= or PRINT
RankVarStdRes	Ranking of the largest variance standardized residuals	RESIDUAL= or PRINT
RawRes	Raw residual matrix	RESIDUAL= or PRINT
SimpleStatistics	Simple statistics, with raw data input	SIMPLE or default
StdErrs	Vector of standard errors	PRIVEC and default
SumSqDif	Sum of squared differences of predetermined elements	PREDET or PALL
tValues	Vector of t values	PRIVEC and default
VarStdRes	Variance of standardized residual matrix	RESIDUAL= or PRINT
WaldTest	Wald test	MODIFICATION or PALL
Weights	Weight matrix	PWEIGHT or PALL
WeightsDet	Determinant of the weight matrix	PWEIGHT or PALL

Table Name	Description	Option
ODS Tables Created by the FACTOR, LINEQS, and RAM Models		
Determination	Coefficients of determination	PDETERM and default
SqMultCorr	Squared multiple correlations	PESTIM or PSHORT
ODS Tables Created by the COSAN and FACTOR Models		
EstParms	Estimated parameter matrix	PESTIM or PSHORT
InitParms	Initial matrix of parameter estimates	PINITIAL or default
ODS Tables Created by the LINEQS and RAM Models		
Indirect Effects	Indirect effects	TOTEFF or PRINT
InitParms	Initial matrix of parameter estimates	PRIMAT and default
LatentScoreCoef	Latent variable regression score coefficients	PLATCOV or PRINT
PredMomentLatent	Predicted latent variable moments	PLATCOV or PRINT
PredMomentManLat	Predicted manifest and latent variable moments	PLATCOV or PRINT
SetCovExog	Set covariance parameters for manifest exogenous variables	PINITIAL or default
Stability	Stability of reciprocal causation	PDETERM and default
StructEq	Variables in the structural equations	PDETERM and default
TotalEffects	Total effects	TOTEFF or PRINT
VarSelection	Manifest variables, if not all are used, selected for modeling	
ODS Tables Created by the FACTOR Model		
FactCorrExog	Correlations among factors	PESTIM or PSHORT
FactScoreCoef	Factor score regression coefficients	PESTIM or PSHORT
RotatedLoadings	Rotated loadings, with ROTATE= option in FACTOR statement	PESTIM or PSHORT
Rotation	Rotation matrix, with ROTATE= option in FACTOR statement	PESTIM or PSHORT
StdLoadings	Standardized factor loadings	PESTIM or PSHORT

Table Name	Description	Option
	ODS Tables Created by the LINEQS Model	
CorrExog	Correlations among exogenous variables	PESTIM or PSHORT
EndogenousVar	Endogenous variables	PESTIM or PSHORT
EstCovExog	Estimated covariances among exogenous variables	PESTIM or PSHORT
EstLatentEq	Estimated latent variable equations	PESTIM or PSHORT
EstManifestEq	Estimated manifest variable equations	PESTIM or PSHORT
EstVarExog	Estimated variances of exogenous variables	PESTIM or PSHORT
ExogenousVar	List of exogenous variables	PESTIM or PSHORT
InCovExog	Input covariances among exogenous variables	PESTIM or PSHORT
InLatentEq	Input latent variable equations	PESTIM or PSHORT
InManifestEq	Input manifest variable equations	PESTIM or PSHORT
InVarExog	Input variances of exogenous variables	PESTIM or PSHORT
StdLatentEq	Standardized latent variable equations	PESTIM or PSHORT
StdManifestEq	Standardized manifest variable equations	PESTIM or PSHORT
	ODS Tables Created by the RAM Model	
InitRAMEstimates	Initial RAM estimates	PESTIM or PSHORT
RAMCorrExog	Correlations among exogenous variables	PESTIM or PSHORT
RAMEstimates	RAM final estimates	PESTIM or PSHORT
RAMStdEstimates	Standardized estimates	PESTIM or PSHORT

Table 12.38 ODS Table Names Produced by the CANCORR Procedure

Table Name	Description	Option
MultStat	Multivariate statistics	

Table Name	Description	Option
ODS Tables Created by PROC CANCORR		
AvgRSquare	Average R-Squares (weighted and unweighted)	VDEP (or WDEP) or SMC (or ALL)
CanCorr	Canonical correlations	
CanStructureVCan	Correlations between the VAR canonical variables and the VAR and WITH variables	Default (unless SHORT)
CanStructureWCan	Correlations between the WITH canonical variables and the WITH and VAR variables	Default (unless SHORT)
ConfidenceLimits	95% confidence limits for the regression coefficients	VDEP (or WDEP) or CLB (or ALL)
Corr	Correlations among the original variables	CORR (or ALL)
CorrRegCoefEst	Correlations among the regression coefficient estimates	VDEP (or WDEP) or CORRB (or ALL)
NObsNVar	Number of observations and variables	SIMPLE (or ALL)
ParCorr	Partial correlations	VDEP (or WDEP) or PCORR (or ALL)
ProbtRegCoef	Prob > |t| for the regression coefficients	VDEP (or WDEP) or PROBT (or ALL)
RawCanCoefV	Raw canonical coefficients for the VAR variables	Default (unless SHORT)
RawCanCoefW	Raw canonical coefficients for the WITH variables	Default (unless SHORT)
RawRegCoef	Raw regression coefficients	VDEP (or WDEP) or B (or ALL)
Redundancy	Canonical redundancy analysis	REDUNDANCY (or ALL)
Regression	Squared multiple correlations and F tests	VDEP (or WDEP) or SMC (or ALL)
SemiParCorr	Semi-partial correlations	VDEP (or WDEP) or SPCORR (or ALL)
SimpleStatistics	Simple statistics	SIMPLE (or ALL)
SqMultCorr	Canonical redundancy analysis: squared multiple correlations	REDUNDANCY (or ALL)
SqParCorr	Squared partial correlations	VDEP (or WDEP) or SQPCORR (or ALL)
SqSemiParCorr	Squared semi-partial correlations	VDEP (or WDEP) or SQSPCORR (or ALL)
StdCanCoefV	Standardized canonical coefficients for the VAR variables	Default (unless SHORT)

Table Name	Description	Option
StdCanCoefW	Standardized canonical coefficients for the WITH variables	Default (unless SHORT)
StdErrRawRegCoef	Standard errors of the raw regression coefficients	VDEP (or WDEP) or SEB (or ALL)
StdRegCoef	Standardized regression coefficients	VDEP (or WDEP) or STB (or ALL)
tValueRegCoef	t values for the regression coefficients	VDEP (or WDEP) or T (or ALL)
ODS Tables Created by the PARTIAL Statement		
CorrOnPartial	Partial correlations	CORR (or ALL)
RSquareRMSEOnPartial	R-Squares and RMSEs on PARTIAL	CORR (or ALL)
StdRegCoefOnPartial	Standardized regression coefficients on PARTIAL	CORR (or ALL)

Table 12.39 ODS Table Names Produced by the CANDISC Procedure

Table Name	Description	Option
ANOVA	Univariate statistics	ANOVA
AveRSquare	Average R-Square	ANOVA
BCorr	Between-class correlations	BCORR
BCov	Between-class covariances	BCOV
BSSCP	Between-class SSCP matrix	BSSCP
BStruc	Between canonical structure	
CanCorr	Canonical correlations	
CanonicalMeans	Class means on canonical variables	
Counts	Number of observations, variables, classes, DF	
CovDF	DF for covariance matrices, not printed	Any *COV option
Dist	Squared distances	MAHALANOBIS
DistFValues	F statistics based on squared distances	MAHALANOBIS

Table Name	Description	Option
DistProb	Probabilities for F statistics from squared distances	MAHALANOBIS
Levels	Class level information	
MultStat	MANOVA	
PCoef	Pooled standard canonical coefficients	
PCorr	Pooled within-class correlations	PCORR
PCov	Pooled within-class covariances	PCOV
PSSCP	Pooled within-class SSCP matrix	PSSCP
PStdMeans	Pooled standardized class means	STDMEAN
PStruc	Pooled within canonical structure	
RCoef	Raw canonical coefficients	
SimpleStatistics	Simple statistics	SIMPLE
TCoef	Total-sample standard canonical coefficients	
TCorr	Total-sample correlations	TCORR
TCov	Total-sample covariances	TCOV
TSSCP	Total-sample SSCP matrix	TSSCP
TSTDMeans	Total standardized class means	STDMEAN
TStruc	Total canonical structure	
WCorr	Within-class correlations	WCORR
WCov	Within-class covariances	WCOV
WSSCP	Within-class SSCP matrices	WSSCP

Table 12.40 ODS Table Names Produced by the CATMOD Procedure

Table Name	Description	Option
	ODS Tables Created by the MODEL Statement	
ANOVA	Analysis of variance	
ConvergenceStatus	Convergence status	ML
CorrB	Correlation matrix of the estimates	CORRB

Table Name	Description	Option
CovB	Covariance matrix of the estimates	COVB
Estimates	Analysis of estimates	Default, unless NOPARM
MaxLikelihood	Maximum likelihood analysis	ML
OneWayFreqs	One-way frequencies	ONEWAY
PopProfiles	Population profiles	Default, unless NOPROFILE
PredictedFreqs	Predicted frequencies	PRED=FREQ
PredictedProbs	Predicted probabilities	PREDICT or PRED=PROB
PredictedValues	Predicted values	PREDICT or PRED=
ResponseCov	Response functions, covariance matrix	COV
ResponseDesign	Response functions, design matrix	WLS, unless NODESIGN
ResponseFreqs	Response frequencies	FREQ
ResponseProbs	Response probabilities	PROB
ResponseProfiles	Response profiles	Default, unless NOPROFILE
XPX	X"*Inv(S)*X matrix	XPX, for WLS
ODS Tables Created by the CONTRAST Statement		
Contrasts	Contrasts	
ContrastEstimates	Analysis of contrasts	ESTIMATE=
ODS Tables Created by the PROC Statement		
DataSummary	Data summary	
ODS Tables Created by the MODEL and LOGLIN Statements		
ResponseMatrix	_RESPONSE_ matrix	Unless NORESPONSE

Table 12.41 ODS Table Names Produced by the CLUSTER Procedure

Table Name	Description	Option
ODS Tables Created by the PROC Statement		
ClusterHistory	Observations or clusters joined, frequencies and other cluster statistics	

Table Name	Description	Option
SimpleStatistics	Simple statistics, before or after trimming	SIMPLE
EigenvalueTable	Eigenvalues of the CORR or COV matrix	

Table 12.42 ODS Table Names Produced by the CORRESP Procedure

Table Name	Description	Option
AdjInGreenacre	Greenacre inertia adjustment	GREENACRE
AdjInBenzecri	Benzecri inertia adjustment	BENZECRI
Binary	Binary table	OBSERVED or BINARY
BinaryPct	Binary table percents	OBSERVED or BINARY
Burt	Burt table	OBSERVED or MCA
BurtPct	Burt table percents	OBSERVED or MCA
CellChiSq	Contributions to Chi Square	CELLCHI2
CellChiSqPct	Contributions, percents	CELLCHI2
ColBest	Col best indicators	
ColContr	Col contributions to inertia	
ColCoors	Col coordinates	
ColProfiles	Col profiles	CP
ColProfilesPct	Col profiles, percents	CP
ColQualMassIn	Col quality, mass, inertia	
ColSqCos	Col squared cosines	
DF	DF, Chi Square (not displayed)	
Deviations	Observed — expected frequencies	DEVIATIONS
DeviationsPct	Observed — expected percentages	DEVIATIONS
Expected	Expected frequencies	EXPECTED
ExpectedPct	Expected percents	EXPECTED
Intertias	Inertia decomposition table	
Observed	Observed frequencies	OBSERVED
ObservedPct	Observed percents	OBSERVED
RowBest	Row best indicators	
RowContr	Row contributions to inertia	

Table Name	Description	Option
RowCoors	Row coordinates	
RowProfiles	Row profiles	RP
RowProfilesPct	Row profiles, percents	RP
RowQualMassIn	Row quality, mass, inertia	
RowSqCos	Row squared cosines	
SupColCoors	Supp col coordinates	
SupColProfiles	Sup col profiles	CP
SupColProfilesPct	Sup col profiles, percents	CP
SupColQuality	Supp col quality	
SupCols	Supplementary col frequency	OBSERVED
SupColsPct	Supplementary col percents	OBSERVED
SupColSqCos	Supplementary col squared cosines	
SupRows	Supplementary row frequencies	OBSERVED
SupRowCoors	Supplementary row coordinates	
SupRowProfiles	Supplementary row profiles	RP
SupRowProfilesPct	Supplementary row profiles, percents	RP
SupRowQuality	Supplementary row quality	
SupRowsPct	Supplementary row percents	OBSERVED
SupRowSqCos	Supplementary row square cosines	

Table 12.43 ODS Table Names Produced by the DISCRIM Procedure

Table Name	Description	Option
ANOVA	Univariate statistics	ANOVA
AvePostCrossVal	Average posterior probabilities, cross validation	POSTERR and CROSSVALIDATE
AvePostResub	Average posterior probabilities, resubstitution	POSTERR
AvePostTestClass	Average posterior probabilities, test classification	POSTERR and TEST=
AveRSquare	Average R-Square	ANOVA
BCorr	Between-class correlations	BCORR

Table Name	Description	Option
BCov	Between-class covariances	BCOV
BSSCP	Between-class SSCP matrix	BSSCP
BStruc	Between canonical structure	CANONICAL
CanCorr	Canonical correlations	CANONICAL
CanonicalMeans	Class means on canonical variables	CANONICAL
ChiSq	Chi-Square information	POOL=TEST
ClassifiedCrossVal	Number of observations and percent classified, cross validation	CROSSVALIDATE
ClassifiedResub	Number of observations and percent classified, resubstitution	
ClassifiedTestClass	Number of observations and percent classified, test classification	TEST=
Counts	Number of observations, variables, classes, DF	
CovDF	DF for covariance matrices, not displayed	Any *COV option
Dist	Squared distances	MAHALONOBIS
DistFValues	F values based on squared distances	MAHALONOBIS
DistGeneralized	Generalized squared distances	
DistProb	Probabilities for F values from squared distances	MAHALONOBIS
ErrorCrossVal	Error count estimates, cross validation	CROSSVALIDATE
ErrorResub	Error count estimates, resubstitution	
ErrorTestClass	Error count estimates, test classification	TEST=
Levels	Class level information	
LinearDiscFunc	Linear discriminant function	POOL=YES
LogDet	Log determinant of the covariance matrix	
MultStat	MANOVA	MANOVA
PCoef	Pooled standard canonical coefficients	CANONICAL
PCorr	Pooled within-class correlations	PCORR
PCov	Pooled within-class covariances	PCOV

Table Name	Description	Option
PSSCP	Pooled within-class SSCP matrix	PSSCP
PStdMeans	Pooled standardized class means	STDMEAN
PStruc	Pooled within canonical structure	CANONICAL
PostCrossVal	Posterior probabilities, cross validation	CROSSLIST or CROSSLISTERR
PostErrCrossVal	Posterior error estimates, cross validation	POSTERR and CROSSVALIDATE
PostErrResub	Posterior error estimates, resubstitution	POSTERR
PostErrTestClass	Posterior error estimates, test classification	POSTERR and TEST=
PostResub	Posterior probabilities, resubstitution	LIST or LISTERR
PostTestClass	Posterior probabilities, test classification	TESTLIST or TESTLISTERR
RCoef	Raw canonical coefficients	CANONICAL
SimpleStatistics	Simple statistics	SIMPLE
TCoef	Total-sample standard canonical coefficients	CANONICAL
TCorr	Total-sample correlations	TCORR
TCov	Total-sample covariances	TCOV
TSSCP	Total-sample SSCP matrix	TSSCP
TStdMeans	Total standardized class means	STDMEAN
TStruc	Total canonical structure	CANONICAL
WCorr	Within-class correlations	WCORR
WCov	Within-class covariances	WCOV
WSSCP	Within-class SSCP matrices	WSSCP

Table 12.44 ODS Table Names Produced by the FACTOR Procedure

Table Name	Description	Option
AlphaCoef	Coefficient alpha for each factor	METHOD=ALPHA
CanCorr	Squared canonical correlations	METHOD=ML

Table Name	Description	Option
CondStdDev	Conditional standard deviations	SIMPLE w/PARTIAL
ConvergenceStatus	Convergence status	METHOD=PRINIT, =ALPHA, =ML, or =ULS
Corr	Correlations	CORR
Eigenvalues	Eigenvalues	Default or SCREE
Eigenvectors	Eigenvectors	EIGENVECTORS
FactorWeightRotate	Factor weights for rotation	HKPOWER=
FactorPattern	Factor pattern	
FactorStructure	Factor structure	ROTATE= any oblique rotation
FinalCommun	Final communalities	Default
FinalCommunWgt	Final communalities with weights	METHOD=ML or METHOD=ALPHA
FitMeasures	Measures of fit	METHOD=ML
ImageCoef	Image coefficients	METHOD=IMAGE
ImageCov	Image covariance matrix	METHOD=IMAGE
ImageFactors	Image factor matrix	METHOD=IMAGE
InputFactorPattern	Input factor pattern	METHOD=PATTERN with PRINT or ALL
InputScoreCoef	Standardized input scoring coefficients	METHOD=SCORE with PRINT or ALL
InterFactorCorr	Inter-factor correlations	ROTATE=any oblique rotation
InvCorr	Inverse correlation matrix	ALL
IterHistory	Iteration history	METHOD=PRINIT, =ALPHA, =ML, or =ULS
MultipleCorr	Squared multiple correlations	METHOD=IMAGE or METHOD=HARRIS
NormObliqueTrans	Normalized oblique transformation matrix	ROTATE=any oblique rotation
ObliqueRotFactPat	Rotated factor pattern	ROTATE=any oblique rotation
ObliqueTrans	Oblique transformation matrix	HKPOWER=
OrthRotFactPat	Rotated factor pattern	ROTATE=any orthogonal rotation
OrthTrans	Orthogonal transformational matrix	ROTATE=any orthogonal rotation
ParCorrControlFactor	Partial correlations controlling factors	RESIDUAL
ParCorrControlVar	Partial correlations controlling other variables	MSA
PartialCorr	Partial correlations	MSA or CORR w/PARTIAL

Table Name	Description	Option
PriorCommunalEst	Prior communality estimates	PRIORS=, METHOD=ML, or METHOD=ALPHA
ProcrustesTarget	Target matrix for Procrustean transformation	ROTATE=PROCRUSTES or ROTATE=PROMAX
ProcrustesTrans	Procrustean transformation matrix	ROTATE=PROCRUSTES or ROTATE=PROMAX
RMSOffDiagPartials	Root mean square off-diagonal partials	RESIDUAL
RMSOffDiagResids	Root mean square off-diagonal residuals	RESIDUAL
ReferenceAxisCorr	Reference axis correlations	ROTATE=any oblique rotation
ReferenceStructure	Reference structure	ROTATE=any oblique rotation
ResCorrUniqueDiag	Residual correlations with uniqueness on the diagonal	RESIDUAL
SamplingAdequacy	Kaiser's measure of sampling adequacy	MSA
SignifTests	Significance tests	METHOD=ML
SimpleStatistics	Simple statistics	SIMPLE
StdScoreCoef	Standardized scoring coefficients	SCORE
VarExplain	Variance explained	
VarExplainWgt	Variance explained with weights	METHOD=ML or METHOD=ALPHA
VarFactorCorr	Squared multiple correlations of the variables with each factor	SCORE
VarWeightRotate	Variable weights for rotation	NORM=WEIGHT or ROTATE=

Table 12.45 ODS Table Names Produced by the FASTCLUS Procedure

Table Name	Description	Option
	ODS Tables Created by the PROC Statement	
ApproxExpOverAllRSq	Approximate expected overall R-Squared, single number	
CCC	Cubic clustering criterion, single number	
ClusterList	Cluster listing, obs, ID, and distances	LIST

Table Name	Description	Option
ClusterSum	Cluster summary, cluster number, distances	PRINTALL
ClusterCenters	Cluster centers	
ClusterDispersion	Cluster dispersion	
ConvergenceStatus	Convergence status	PRINTALL
Criterion	Criterion based on final seeds, single number	
DistBetweenClust	Distance between clusters	
InitialSeeds	Initial seeds	
IterHistory	Iteration history, various statistics for each iteration	PRINTALL
MinDist	Minimum distance between initial seeds, single number	PRINTALL
NumberOfBins	Number of bins	
ObsOverAllRSquare	Observed overall R-Squared. single number	SUMMARY
PrelScaleEst	Preliminary L(1) scale estimate, single number	PRINTALL
PseudoFStat	Pseudo F statistic, single number	
SimpleStatistics	Simple statistics for input variables	
VariableStat	Statistics for variables within clusters	

Table 12.46 ODS Table Names Produced by the GAM Procedure

Table Name	Description	Option
	ODS Tables Created by the PROC Statement	
ANODEV	Analysis of deviance table for smoothing variables	
ClassSummary	Summary of class variables	
InputSummary	Data summary	
IterSummary	Iteration summary	
FitSummary	Fit parameters and fit summary	

Table Name	Description	Option
ParameterEstimates	Parameter estimation for regression variables	
ODS Tables Created by the MODEL Statement		
Iteration	Iteration history table	ITPRINT

Table 12.47 ODS Table Names Produced by the GENMOD Procedure

Table Name	Description	Option
ODS Tables Created by the CLASS Statement		
ClassLevels	Class variable levels	
ODS Tables Created by the CONTRAST Statement		
Contrasts	Tests of contrasts	
ContrastCoef	Contrast coefficients	E
LinDep	Linearly dependent rows of contrasts	
NonEst	Nonestimable rows of contrasts	
ODS Tables Created by the MODEL Statement		
ConvergenceStatus	Convergence status	
CorrB	Parameter estimate correlation matrix	CORRB
CovB	Parameter estimate covariance matrix	COVB
IterLRCI	Iteration history for likelihood ratio confidence intervals	LRCI ITPRINT
IterParms	Iteration history for parameter estimates	ITPRINT
IterType3	Iteration history for Type 3 statistics	TYPE3 ITPRINT
LRCI	Likelihood ratio confidence intervals	LRCI ITPRINT
LagrangeStatistics	Lagrange statistics	NOINT or NOSCALE
LastGradHess	Last evaluation of the gradient and Hessian	ITPRINT
ModelInfo	Model information	

Table Name	Description	Option
Modelfit	Goodness-of-fit statistics	
ObStats	Observation-wise statistics	OBSTATS, CL, PREDICTED, RESIDUALS, or XVARS
ParameterEstimates	Parameter estimates	
ParmInfo	Parameter indices	
ResponseProfiles	Frequency counts for multinomial models	DIST=MULTINOMIAL
Type1	Type 1 tests	TYPE1
Type3	Type 3 tests	TYPE3
ODS Tables Created by the ESTIMATE Statement		
Estimates	Estimates of contrasts	
EstimateCoef	Contrast coefficients	E
ODS Tables Created by the REPEATED Statement		
GEEEmpPEst	GEE parameter estimates with empirical standard errors	
GEELogORInfo	GEE log odds ratio model information	LOGOR=
GEEModInfo	GEE model information	
GEEModPEst	GEE parameter estimates with model-based standard errors	MODELSE
GEENCorr	GEE model-based correlation matrix	MCORRB
GEENCov	GEE model-based covariance matrix	MCOVB
GEERCorr	GEE empirical correlation matrix	ECORRB
GEERCov	Gee empirical covariance matrix	ECOVB
GEEWCorr	GEE working correlation matrix	CORRW
ODS Tables Created by the MODEL CONTRAST Statement		
IterContrasts	Iteration history for contrasts	ITPRINT
ODS Tables Created by the MODEL REPEATED Statement		
IterParmsGEE	Iteration history for GEE parameter estimates	ITPRINT

Table Name	Description	Option
LastGEEGrad	Last evaluation of the generalized gradient and Hessian	ITPRINT
ODS Tables Created by the LSMEANS Statement		
LSMeanCoef	Coefficients for least squares means	E
LSMeanDiffs	Least squares means differences	DIFF
LSMeans	Least squares means	

Table 12.48 ODS Table Names Produced by the GLM Procedure

Table Name	Description	Option
DependentInfo	Simultaneously analyzed dependent variables	Default when there are multiple dependent variables with different patterns of missing values
FitStatistics	R-Square, C.V., root MSE, and dependent mean	
MatrixRepresentation	X matrix element representation	As needed for other options
ModelANOVA	ANOVA for model terms	
NObs	Number of observations	
OverallANOVA	Overall ANOVA	
ODS Tables Created by the CLASS Statement		
ClassLevels	Classification variable levels	
ODS Tables Created by the CONTRAST Statement		
AltErrContrasts	ANOVA table for contrasts with alternative error	E=
ContrastCoef	L matrix for contrast	EST
Contrasts	ANOVA table for contrasts	
ODS Tables Created by the ESTIMATE Statement		
Estimates	Estimate statement result	

Table Name	Description	Option
	ODS Tables Created by the LSMEANS Statement	
Diff	PDiff matrix of least-squares means	PDIFF
LSMeanCL	Confidence interval for LS-means	CL
LSMeanCoef	Coefficients of least-squares means	E
LSMeanDiffCL	Confidence interval for LS-mean differences	PDIFF and CL
LSMeans	Least-squares means	
SimDetails	Details of difference quantile simulation	ADJUST=SIMULATE(REPORT)
SimResults	Evaluation of difference quantile simulation	ADJUST=SIMULATE(REPORT)
SlicedANOVA	Sliced effect ANOVA table	SLICE
	ODS Tables Created by the MEANS Statement	
Bartlett	Bartlett's homogeneity of variance test	HOVTEST=BARTLETT
CLDiffs	Multiple comparisons of pairwise differences	CLDIFF, DUNNETT, or (Unequal cells and not LINES)
CLDiffsInfo	Information for multiple comparisons of pairwise differences	CLDIFF, DUNNETT, or (Unequal cells and not LINES)
CLMeans	Multiple comparisons of means with confidence/comparison interval	CLM
CLMeansInfo	Information for multiple comparison of means with confidence/comparison interval	CLM
HOVFTest	Homogeneity of variance ANOVA	HOVTEST
MCLines	Multiple comparisons LINES output	LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (Equal cells and not CLDIFF)
MCLinesInfo	Information for multiple comparison LINES output	LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (Equal cells and not CLDIFF)

Table Name	Description	Option
MCLinesRange	Ranges for multiple range MC tests	LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (Equal cells and not CLDIFF)
Means	Group means	
Welch	Welch's ANOVA	WELCH
ODS Tables Created by the MODEL Statement		
Aliasing	Type 1, 2, 3, 4 aliasing structure	(E1, E2, E3, or E4) and ALIASING
EstFunc	Type 1, 2, 3, 4 estimable functions	E1, E2, E3, or E4
GAliasing	General form of aliasing structure	E and ALIASING
GEstFunc	General form of estimable functions	E
InvXPX	Inv(X"X) matrix	INVERSE
ParameterEstimates	Estimated linear model coefficients	SOLUTION
PredictedInfo	Predicted values info	PREDICTED, CLM, or CLI
PredictedValues	Predicted values	PREDICTED, CLM, or CLI
Tolerances	X"X tolerances	TOLERANCE
XPX	X"X matrix	XPX
ODS Tables Created by the MANOVA or REPEATED Statements		
CanAnalysis	Canonical analysis	CANONICAL
CanCoef	Canonical coefficients	CANONICAL
CanStructure	Canonical structure	CANONICAL
ErrorSSCP	Error SSCP matrix	PRINTE
HypothesisSSCP	Hypothesis SSCP matrix	PRINTH
PartialCorr	Partial correlation matrix	PRINTE
ODS Tables Created by the MANOVA Statement		
CharStruct	Characteristic roots and vectors	Not CANONICAL
MANOVATransform	Multivariate transformation matrix	M=
MultStat	Multivariate tests	

Table Name	Description	Option
Tests	Summary ANOVA for specified MANOVA H= effects	H=SUMMARY
	ODS Tables Created by the RANDOM Statement	
ExpectedMeanSquares	Expected mean squares	
QForm	Quadratic form for expected mean squares	Q
RandomModelANOVA	Random effect tests	TEST
	ODS Tables Created by the REPEATED Statement	
CharStruct	Characteristic roots and vectors	PRINTRV
Epsilons	Greenhouse-Geisser and Huynh-Feldt epsilons	
RepeatedLevelInfo	Correspondence between dependents and repeated measures levels	
RepeatedTransform	Repeated measures transformation matrix	PRINTM
Sphericity	Sphericity tests	PRINTE
	ODS Tables Created by the TEST Statement	
AltErrTests	ANOVA table for tests with alternative error	E=

Table 12.49 ODS Table Names Produced by the GLMMOD Procedure

Table Name	Description	Option
DependentInfo	Simultaneously analyzed dependent variables	Default when there are multiple dependent variables
DesignPoints	Design matrix	
NObs	Number of observations	
Parameters	Parameters and associated column numbers	
	ODS Tables Created by the CLASS Statement	
ClassLevels	Table of class levels	

Table 12.50 ODS Table Names Produced by the GLMPOWER Procedure

Table Name	Description	Option
FixedElements	Factoid with single-valued analysis parameters	Default
Output	All input and computed analysis parameters, error messages, and information messages for each scenario	Default
PlotContent	Data contained in plots, including analysis parameters and indices identifying plot features. (Note: This table is saved as a dataset and not displayed in PROC GLMPOWER output.)	PLOT

Table 12.51 ODS Table Names Produced by the INBREED Procedure

Table Name	Description	Option
ODS Tables Created by the GENDER Statement		
AvgCovCoef	Averages of covariance coefficient matrix	COVAR and AVERAGE
AvgInbreedingCoef	Averages of inbreeding coefficient matrix	AVERAGE
ODS Tables Created by the MATINGS Statement		
MatingCovCoef	Covariance coefficients of matings	COVAR
MatingInbreedingCoef	Inbreeding coefficients of matings	
ODS Tables Created by the PROC Statement		
CovarianceCoefficient	Covariance coefficient table	COVAR
InbreedingCoefficient	Inbreeding coefficient table	
IndividualCovCoef	Inbreeding coefficients of individuals	IND and COVAR

Table Name	Description	Option
IndividualInbreedingCoef	Inbreeding coefficients of individuals	IND
NumberOfObservations	Number of observations	

Table 12.52 ODS Table Names Produced by the KDE Procedure

Table Name	Description
BivariateStatistics	Bivariate statistics
Controls	Control variables
Inputs	Input information
Levels	Levels of density estimate
Percentiles	Percentiles of data
Statistics	Basic statistics

Table 12.53 ODS Table Names Produced by the LATTICE Procedure

Table Name	Description
ANOVA	Analysis of variance
AdjTreatmentMeans	Adjusted treatment means
Statistics	Additional statistics

Table 12.54 ODS Table Names Produced by the LIFEREG Procedure

Table Name	Description	Option
	ODS Tables Created by the CLASS Statement	
ClassLevels	Class variable levels	
	ODS Tables Created by the MODEL Statement	
ConvergenceStatus	Convergence status	

Table Name	Description	Option
CorrB	Parameter estimate correlation matrix	CORRB
CovB	Parameter estimate covariance matrix	COVB
IterHistory	Iteration history	ITPRINT
LagrangeStatistics	Lagrange statistics	NOINT or NOSCALE
LastGrad	Last evaluation of the gradient	ITPRINT
LastHess	Last evaluation of the Hessian	ITPRINT
ModelInfo	Model information	
ParameterEstimates	Parameter estimates	
ParmInfo	Parameter indices	
Type3Analysis	Type 3 tests	
ODS Tables Created by the PROBPLOT Statement		
EMIterHistory	Iteration history for Turnbull algorithm	ITPRINTEM
ProbEstimates	Nonparametric CDF estimates	PPOUT
Turnbull	Probability estimates from Turnbull algorithm	ITPRINTEM

Table 12.55 ODS Table Names Produced by the LIFETEST Procedure

Table Name	Description	Option
ODS Tables Created by the PROC Statement		
CensorPlot	Line-printer plot of censored observations	PLOT=(C, S, LS or LLS), METHOD=PL, and LINEPRINTER
CensoredSummary	Number of event and censored observations	METHOD=PL
DensityPlot	Plot of the density	PLOT=(D) and METHOD=LT
HazardPlot	Plot of the hazards function	PLOT=(H) and METHOD=LT
LifetableEstimates	Lifetable survival estimates	METHOD=LT
LogLogSurvivalPlot	Plot of the log of the negative log survivor function	PLOT=(LLS)
LogSurvivalPlot	Plot of the log survivor function	PLOT=(LS)

Table Name	Description	Option
Means	Mean and standard error of survival times	METHOD=PL
ProductLimitEstimates	Product-limit survival estimates	METHOD=PL
Quartiles	Quartiles of the survival distribution	METHOD=PL
SurvivalPlot	Plot of the survivor function	PLOT=(S)
ODS Tables Created by the STRATA Statement		
HomStats	Rank statistics for testing strata homogeneity	
HomTests	Tests for strata homogeneity	
LogHomCov	Covariance matrix for the log-rank statistics for strata homogeneity	
WilHomCov	Covariance matrix for the Wilcoxon statistics for strata homogeneity	
ODS Tables Created by the TEST Statement		
LogForStepSeq	Forward stepwise sequence for the log-rank statistics for association	
LogTestCov	Covariance matrix for log-rank statistics for association	
LogUniChisq	Univariate Chi-Squares for log-rank statistic for association	
WilForStepSeq	Forward stepwise sequence for the log-rank statistics for association	
WilTestCov	Covariance matrix for log-rank statistics for association	
WilUniChiSq	Univariate Chi-Squares for Wilcoxon statistic for association	

Table 12.56 ODS Table Names Produced by the LOESS Procedure

Table Name	Description	Option
FitSummary	Specified fit parameters and fit summary	
ScaleDetails	Extent and scaling of the independent variables	
ODS Tables Created by the MODEL Statement		
kdTree	Structure of kd tree used	DETAILS(kdTree)
ModelSummary	Summary of all models evaluated	DETAILS(ModelSummary)
OutputStatistics	Coordinates and fit results at input data points	DETAILS(OutputStatistics)
PredAtVertices	Coordinates and fitted values at kd tree vertices	DETAILS(PredAtVertices)
SmoothingCriterion	Criterion value and selected smoothing parameter	SELECT
ODS Tables Created by the SCORE Statement		
ScoreResults	Coordinates and fit results at scoring points	PRINT

Table 12.57 ODS Table Names Produced by the LOGISTIC Procedure

Table Name	Description	Option
ODS Tables Created by the CONTRAST Statement		
ContrastCoeff	L matrix from CONTRAST	E
ContrastEstimate	Estimates from CONTRAST	ESTIMATE=
ContrastTest	Wald test for CONTRAST	
ODS Tables Created by the EXACT Statement		
ExactOddsRatio	Exact odds ratio	ESTIMATE=ODDS or ESTIMATE=BOTH
ExactParmEst	Parameter estimates	ESTIMATE, ESTIMATE=PARM, or ESTIMATE=BOTH

Table Name	Description	Option
ExactTests	Conditional exact tests	
SuffStats	Sufficient statistics	OUTDIST=
	ODS Tables Created by the MODEL Statement	
Association	Association of predicted probabilities and observed responses	Default
BestSubsets	Best subset selection	SELECTION=SCORE
ClassLevelInfo	CLASS variable levels and design variables	Default (with CLASS variables)
Classification	Classification table	CTABLE
CLOddsPL	Profile likelihood confidence limits for odds ratios	CLODDS=PL
CLOddsWald	Wald's confidence limits for odds ratios	CLODDS=WALD
CLParmPL	Profile likelihood confidence limits for parameters	CLPARM=PL
CLParmWald	Wald's confidence limits for parameters	CLPARM=WALD
ConvergenceStatus	Convergence status	Default
CorrB	Estimated correlation matrix of parameter estimators	CORRB
CovB	Estimated covariance matrix of parameter estimators	COVB
CumulativeModelTest	Test of the cumulative model assumption	(Ordinal response)
EffectNotInModel	Test for effects not in model	SELECTION=S or F
FastElimination	Fast backward elimination	SELECTION=B, FAST
FitStatistics	Model fit statistics	Default
GlobalScore	Global score test	NOFIT
GlobalTests	Test for global null hypothesis	Default
GoodnessOfFit	Pearson and deviance goodness-of-fit tests	SCALE
IndexPlots	Batch capture of the index plots	IPLOTS
Influence	Regression diagnostics	INFLUENCE
IterHistory	Iteration history	ITPRINT
LackFitChiSq	Hosmer-Lemeshow Chi-Square test results	LACKFIT
LackFItPartition	Partition for the Hosmer-Lemeshow test	LACKFIT

Table Name	Description	Option
LastGradient	Last evaluation of gradient	ITPRINT
LogLikeChange	Final change in the log likelihood	ITPRINT
ModelBuildingSummary	Summary of model building	SELECTION=B, F, or S
OddsRatios	Odds ratios	Default
ParameterEstimates	Maximum likelihood estimates of model parameters	Default
RSquare	R-Square	RSQUARE
ResidualChiSq	Residual Chi-Square	SELECTION=F or B
Type3	Type 3 tests of effects	Default (with CLASS variables)
ODS Tables Created by the ODSRATIOS Statement		
OddsRatiosWald	Odds ratios with Wald confidence limits	CL=WALD
OddsRatiosPL	Odds ratios with PL confidence limits	CL=PL
ODS Tables Created by the PROC Statement		
ClassFreq	Frequency breakdown of CLASS variables	SIMPLE
ClassWgt	Weight breakdown of CLASS variables	SIMPLE
ModelInfo	Model information	Default
ResponseProfile	Response profile	Default
SimpleStatistics	Summary statistics for explanatory variables	SIMPLE
ODS Tables Created by the STRATA Statement		
StrataSummary	Number of strata with specific response frequencies	Default
StrataInfo	Event and non-event frequencies for each stratum	INFO
ODS Tables Created by the TEST Statement		
TestPrint1	L[cov(b)]L" and Lb-c	PRINT
TestPrint2	Ginv(L[cov(b)]L") and Ginv(L[cov(b)]L")(Lb-c)	PRINT
TestStmts	Linear hypothesis testing results	Default

Table Name	Description	Option
ODS Tables Created by the WEIGHT Statement		
ClassWgt	Weight breakdown of CLASS variables	SIMPLE

Table 12.58 ODS Table Names Produced by the MDS Procedure

Table Name	Description	Option
ConvergenceStatus	Convergence status	
DimensionCoef	Dimension coefficients	PCOEF w/COEF= not IDENTITY
FitMeasures	Measures of fit	PFIT
IterHistory	Iteration history	
PConfig	Estimated coordinates of the objects in the configuration	PCONFIG
PData	Data matrices	PDATA
PInAvData	Initial sum of weights and weighted average of data matrices with INAV=DATA	PINAVDATA
PInEigval	Initial eigenvalues	PINEIGVAL
PInEigvec	Initial eigenvectors	PINEIGVEC
PInWeight	Initialization weights	PINWEIGHT
Transformations	Transformation parameters	PTRANS w/LEVEL=RATIO, INTERVAL, or LOGINTERVAL

Table 12.59 ODS Table Names Produced by the MI Procedure

Table Name	Description	Option
Corr	Pairwise correlations	SIMPLE
MissPattern	Missing data patterns	
ModelInfo	Model information	
ParameterEstimates	Parameter estimates	
Univariate	Univariate statistics	SIMPLE

Table Name	Description	Option
VarianceInfo	Between, within, and total variances	
	ODS Tables Created by the EM Statement	
EMEstimates	EM (MLE) estimates	
EMInitEstimates	EM initial estimates	
EMIterHistory	EM (MLE) iteration history	ITPRINT
	ODS Tables Created by the MCMC Statement	
EMPostEstimates	EM (posterior mode) estimates	INITIAL=EM
EMPostIterHistory	EM (posterior mode) iteration history	INITIAL=EM (ITPRINT)
EMWLF	Worst linear function	WLF
MCMCInitEstimates	MCMC initial estimates	DISPLAYINIT
	ODS Tables Created by the MONOTONE Statement	
MonoDiscrim	Discriminant model group means	DISCRIM (/DETAILS)
MonoLogistic	Logistic model	LOGISTIC (/DETAILS)
MonoModel	Multiple monotone models	
MonoPropensity	Propensity score model logistic function	PROPENSITY (/DETAILS)
MonoReg	Regression model	REG (/DETAILS)
MonoRegPPM	Predicted mean matching model	REGPMM (/DETAILS)
	ODS Tables Created by the TRANSFORM Statement	
Transform	Variable transformations	

Table 12.60 ODS Table Names Produced by the MIANALYZE Procedure

Table Name	Description	Option
BCov	Between-imputation covariance matrix	BCOV
ModelInfo	Model information	
MultStat	Multivariate inference	MULT

Table Name	Description	Option
ParameterEstimates	Parameter estimates	
TCov	Total covariance matrix	TCOV
VarianceInfo	Variance information	
WCov	Within-imputation covariance matrix	WCOV
	ODS Tables Created by the TEST Statement	
TestBCov	Between-imputation covariance matrix for Lβ	BCOV
TestMultStat	Multivariate inference for Lβ	MULT
TestParameterEstimates	Parameter estimates for Lβ	
TestSpec	Test specification, L and c	
TestTCov	Total covariance matrix for Lβ	TCOV
TestVarianceInfo	Variance information for Lβ	
TestWCov	Within—imputation covariance matrix for Lβ	WCOV

Table 12.61 ODS Table Names Produced by the MIXED Procedure

Table Name	Description	Option
AccRates	Acceptance rates for posterior sampling	PRIOR
AsyCorr	Asymptotic correlation matrix of covariance parameters	PROC MIXED ASYCORR
AsyCov	Asymptotic covariance matrix of covariance parameters	PROC MIXED ASYCOV
Base	Base densities used for posterior sampling	PRIOR
Bound	Computed bound for posterior rejection sampling	PRIOR
CholG	Cholesky root of the estimated G matrix	RANDOM / GC
CholR	Cholesky root of blocks of the estimated R matrix	REPEATED / RC
CholV	Cholesky root of blocks of the estimated V matrix	RANDOM / VC

Table Name	Description	Option
ClassLevels	Level information from the CLASS statement	Default output
Coef	L matrix coefficients	E option on MODEL, CONTRAST, ESTIMATE, or LSMEANS
Contrasts	Results from the CONTRAST statements	CONTRAST
ConvergenceStatus	Convergence status	Default
CorrB	Approximate correlation matrix of fixed-effects parameter estimates	MODEL / CORRB
CovB	Approximate covariance matrix of fixed-effects parameter estimates	MODEL / COVB
CovParms	Estimated covariance parameters	Default output
Diffs	Differences of LS-means	LSMEANS / DIFF (or PDIFF)
Dimensions	Dimensions of the model	Default output
Estimates	Results from ESTIMATE statements	ESTIMATE
FitStatistics	Fit statistics	Default
G	Estimated G matrix	RANDOM / G
GCorr	Correlation matrix from the estimated G matrix	RANDOM / GCORR
HLM1	Type 1 Hotelling-Lawley-McKeon tests of fixed effects	MODEL / HTYPE=1 and REPEATED / HLM TYPE=UN
HLM2	Type 2 Hotelling-Lawley-McKeon tests of fixed effects	MODEL / HTYPE=2 and REPEATED / HLM TYPE=UN
HLM3	Type 3 Hotelling-Lawley-McKeon tests of fixed effects	REPEATED / HLM TYPE=UN
HLPS1	Type 1 Hotelling-Lawley-Pillai-Samson tests of fixed effects	MODEL / HTYPE=1 and REPEATED / HLPS TYPE=UN
HLPS2	Type 2 Hotelling-Lawley-Pillai-Samson tests of fixed effects	MODEL / HTYPE=1 and REPEATED / HLPS TYPE=UN
HLPS3	Type 3 Hotelling-Lawley-Pillai-Samson tests of fixed effects	REPEATED / HLPS TYPE=UN
Influence	Influence diagnostics	MODEL / INFLUENCE
InfoCrit	Information criteria	PROC MIXED IC

Table Name	Description	Option
InvCholG	Inverse Cholesky root of the estimated G matrix	RANDOM / GCI
InvCholR	Inverse Cholesky root of blocks of the estimated R matrix	REPEATED / RCI
InvCholV	Inverse Cholesky root of blocks of the estimated V matrix	RANDOM / VCI
InvCovB	Inverse of approximate covariance matrix of fixed-effects parameter estimates	MODEL / COVBI
InvG	Inverse of the estimated G matrix	RANDOM / GI
InvR	Inverse of blocks of the estimated R matrix	REPEATED / RI
InvV	Inverse of blocks of the estimated V matrix	RANDOM / VI
IterHistory	Iteration history	Default output
LComponents	single degree of freedom estimates corresponding to rows of the L matrix for fixed effects	MODEL / LCOMPONENTS
LRT	Likelihood ratio test	Default output
LSMeans	LS-means	LSMEANS
MMEq	Mixed model equations	PROC MIXED MMEQ
MMEqSol	Mixed model equations solution	PROC MIXED MMEQSOL
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
ParmSearch	Parameter search values	PARMS
Posterior	Posterior sampling information	PRIOR
R	Blocks of the estimated R matrix	REPEATED / R
RCorr	Correlation matrix from blocks of the estimated R matrix	REPEATED / RCORR
Search	Posterior density search table	PRIOR / PSEARCH
Slices	Tests of LS-means slices	LSMEANS / SLICE=
SolutionF	Fixed effects solution vector	MODEL / S
SolutionR	Random effects solution vector	RANDOM / S
Tests1	Type 1 tests of fixed effects	MODEL / HTYPE=1
Tests2	Type 1 tests of fixed effects	MODEL / HTYPE=2
Tests3	Type 1 tests of fixed effects	Default output

Table Name	Description	Option
Type1	Type 1 analysis of variance	PROC MIXED METHOD=TYPE1
Type2	Type 2 analysis of variance	PROC MIXED METHOD=TYPE2
Type3	Type 3 analysis of variance	PROC MIXED METHOD=TYPE3
Trans	Transformation of covariance parameters	PRIOR / PTRANS
V	Blocks of the estimated V matrix	RANDOM / V
VCorr	Correlation matrix from blocks of the estimated V matrix	RANDOM / VCORR

Table 12.62 ODS Table Names Produced by the MODECLUS Procedure

Table Name	Description	Option
	ODS Tables Created by the PROC Statement	
BoundaryFreq	Boundary objects information	BOUNDARY (or ALL)
ClusterList	Cluster listing, cluster ID, frequency, density, etc.	LIST (or ALL)
ClusterStats	Cluster statistics	
ClusterStats	Cluster statistics, significance test statistics	TEST or JOIN (or ALL)
ClusterSummary	Cluster summary	
ClusterSummary	Cluster summary, crossvalidation criterion	CROSS or CROSSLIST (or ALL)
ClusterSummary	Cluster summary, clusters joined information	JOIN (or ALL)
CrossList	Cross-validated log density	CROSSLIST
ListLocal	Local dimensionality estimates	LOCAL
Neighbor	Nearest neighbor list	NEIGHBOR (or ALL)
SimpleStatistics	Simple statistics	SIMPLE (or ALL)
Trace	Trace of clustering algorithm (METHOD=6 only)	TRACE (or ALL) with METHOD=6
UnassignObjects	Information on unassigned objects	LIST (or ALL)

Table 12.63 ODS Table Names Produced by the MULTTEST Procedure

Table Name	Description	Option
Continuous	Continuous variable tabulations	TEST with MEAN
Contrasts	Contrast coefficients	
Discrete	Discrete variable tabulations	TEST with CA, FT, PETO, or FISHER
ModelInfo	Model information	
pValues	p-values from the tests	

Table 12.64 ODS Table Names Produced by the NESTED Procedure

Table Name	Description
ANCOVA	Analysis of covariance
ANOVA	Analysis of variance
EMSCoef	Coefficients of expected mean squares
Statistics	Overall statistics for fit

Table 12.65 ODS Table Names Produced by the NLIN Procedure

Table Name	Description
ANOVA	Analysis of variance
ConvergenceStatus	Convergence status
CorrB	Correlation of the parameters
EstSummary	Summary of the estimation
IterHistory	Iteration output
MissingValues	Missing values generated by the program
ParameterEstimates	Parameter estimates
ODS Tables Created by the LIST Statement	
ProgList	List of the compiled program

Table Name	Description
ODS Tables Created by the LISTCODE Statement	
CodeList	List of program statements
ODS Tables Created by the LISTDEP Statement	
CodeDependency	Variable cross reference
ODS Tables Created by the LISTDER Statement	
FirstDerivatives	First derivative table

Table 12.66 ODS Table Names Produced by the NLMIXED Procedure

Table Name	Description	Option
AdditionalEstimates	Results from ESTIMATE statements	ESTIMATE
ConvergenceStatus	Convergence status	
CorrMatAddEst	Correlation matrix of additional estimates	ECORR
CorrMatParmEst	Correlation matrix of parameter estimates	CORR
CovMatAddEst	Covariance matrix of additional estimates	ECOV
CovMatParmEst	Covariance matrix of parameter estimates	COV
DerAddEst	Derivatives of additional estimates	EDER
Dimensions	Dimensions of the problem	
FitStatistics	Fit statistics	
Hessian	Second derivative matrix	HESS
IterHistory	Iteration history	
Parameters	Parameters	
ParameterEstimates	Parameter estimates	
Specifications	Model specifications	
StartingHessian	Starting hessian matrix	START HESS
StartingValues	Starting values and gradient	START

Table 12.67 ODS Table Names Produced by the NPAR1WAY Procedure

Table Name	Description	Option
	ODS Tables Created by the EXACT Statement	
ABMC	Monte Carlo estimates for the Ansari-Bradley exact test	AB or MC
DataScoresMC	Monte Carlo estimates for the exact test based on data scores	SCORES=DATA or MC
KlotzMC	Monte Carlo estimates for the Klotz exact test	KLOTZ or MC
KolSmirExactTest	Kolmogorov-Smirnov exact test	KS
KruskalWallisMC	Monte Carlo estimates for the Kruskal-Wallis exact test	WILCOXON or MC
KSMC	Monte Carlo estimates for the Kolmogorov-Smirnov exact test	KS or MC
MedianMC	Monte Carlo estimates for the median exact test	MEDIAN or MC
MoodMC	Monte Carlo estimates for the Mood exact test	MOOD or MC
SavageMC	Monte Carlo estimates for the Savage exact test	SAVAGE or MC
STMC	Monte Carlo estimates for the Siegel-Tukey one-way analysis	ST or MC
VWMC	Monte Carlo estimates for the Van der Waerden exact test	VW or MC
WilcoxonMC	Monte Carlo estimates for the Wilcoxon two-sample exact test	WILCOXON or MC
	ODS Tables Created by the PROC Statement	
ANOVA	Analysis of variance	ANOVA
ABAnalysis	Ansari-Bradley one-way analysis	AB
ABScores	Ansari-Bradley scores	AB
ABTest	Ansari-Bradley two-sample test	AB
ClassMeans	Class means	ANOVA
CVMStats	Cramer-von Mises statistics	EDF
CVMTest	Cramer-von Mises test	EDF
DataScores	Data scores	SCORES=DATA
DataScoresAnalysis	Data scores one-way analysis	SCORES=DATA
DataScoresTest	Data scores two-sample test	SCORES=DATA

Table Name	Description	Option
KlotzAnalysis	Klotz one-way analysis	KLOTZ
KlotzScores	Klotz scores	KLOTZ
KlotzTest	Klotz two-sample test	KLOTZ
KolSmir2Stats	Kolmogorov-Smirnov two-sample statistics	EDF
KolSmirStats	Kolmogorov-Smirnov statistics	EDF
KolSmirTest	Kolmogorov-Smirnov test	EDF
KruskalWallisTest	Kruskal-Wallis test	WILCOXON
KuiperStats	Kuiper two-sample statistics	EDF
KuiperTest	Kuiper test	EDF
MedianAnalysis	Median one-way analysis	MEDIAN
MedianScores	Median scores	MEDIAN
MedianTest	Median two-sample test	MEDIAN
MoodAnalysis	Mood one-way analysis	MOOD
MoodScores	Mood scores	MOOD
MoodTest	Mood two-sample test	MOOD
SavageAnalysis	Savage one-way analysis	SAVAGE
SavageScores	Savage scores	SAVAGE
SavageTest	Savage two-sample test	SAVAGE
STAnalysis	Siegel-Tukey one-way analysis	ST
STScores	Siegel-Tukey scores	ST
STTest	Siegel-Tukey two-sample test	ST
VWAnalysis	Van der Waerden one-way analysis	VW
VWScores	Van der Waerden scores	VW
VWTest	Van der Waerden two-sample test	VW
WilcoxonScores	Wilcoxon scores	WILCOXON
WilcoxonTest	Wilcoxon two-sample test	WILCOXON

Table 12.68 ODS Table Names Produced by the ORTHOREG Procedure

Table Name	Description
ANOVA	Analysis of variance
FitStatistics	Overall statistics for fit

Table Name	Description
ParameterEstimates	Parameter estimates
ODS Tables Created by the CLASS Statement	
Levels	Table of class levels

Table 12.69 ODS Table Names Produced by the PPHREG Procedure

Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
BestSubsets	Best subset selection	SELECTION=SCORE
CensoredSummary	Summary of event and censored observations	
ConvergenceStatus	Convergence status	
CorrB	Estimated correlation matrix of parameter estimators	CORRB
CovB	Estimated covariance matrix of parameter estimators	COVB
FitStatistics	Model fit statistics	
GlobalScore	Global Chi-Square test	NOFIT
GlobalTests	Tests of the global null hypothesis	
IterHistory	Iteration history	ITPRINT
LastGradient	Last evaluation of gradient	ITPRINT
ModelBuildingSummary	Summary of model building	SELECTION=B, F, or S
ParameterEstimates	Maximum likelihood estimates of model parameters	
ResidualChiSq	Residual Chi-Square	SELECTION=F or B
VariablesNotInModel	Analysis of variables not in the model	SELECTION=F or S
ODS Tables Created by the PROC Statement		
ModelInfo	Model information	
SimpleStatistics	Summary statistics for explanatory variables	SIMPLE
ODS Tables Created by the TEST Statement		

Table Name	Description	Option
TestAverage	Average effect for test	AVERAGE
TestCoeff	Coefficients for linear hypothesis	E
TestPrint1	L[cov(b)]L" and Lb-c	PRINT
TestPrint2	Ginv(L[cov(b)]L") and Ginv(L[cov(b)]L")(Lb-c)	PRINT
TestStmts	Linear hypotheses testing results	

Table 12.70 ODS Table Names Produced by the PLAN Procedure

Table Name	Description
Plan	Computed plan
ODS Tables Created by the FACTOR and TREATMENT Statements	
PFInfo	Plot factor information
TFInfo	Treatment factor information
ODS Tables Created by the FACTOR and no TREATMENT Statements	
FInfo	General factor information

Table 12.71 ODS Table Names Produced by the PLS Procedure

Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
CenScaleParms	Parameter estimates for centered and scaled data	SOLUTION
ParameterEstimates	Parameter estimates for raw data	SOLUTION
ODS Tables Created by the PROC Statement		
CVResults	Results of cross validation	CV
CodedCoef	Coded coefficients	DETAILS

Table Name	Description	Option
PercentVariation	Variation accounted for by each factor	
ResidualSummary	Residual summary from cross validation	CV
XEffectCenScale	Centering and scaling information for predictor effects	CENSCALE
XLoadings	Loadings for independents	DETAILS
XVariableCenScale	Centering and scaling information for predictor effects	CENSCALE and VARSCALE
XWeights	Weights for independents	DETAILS
YVariableCenScale	Centering and scaling information for responses	CENSCALE
YWeights	Weights for dependents	DETAILS

Table 12.72 ODS Table Names Produced by the POWER Procedure

Table Name	Description	Option
FixedElements	Factoid with single-valued analysis parameters	Default
Output	All input and computed analysis parameters, error messages, and information messages for each scenario	Default
PlotContent	Data contained in plots, including analysis parameters and indices identifying plot features. (Note: This table is saved as a dataset and not displayed in PROC POWER output.)	PLOT

Table 12.73 ODS Table Names Produced by the PRINCOMP Procedure

Table Name	Description	Option
Corr	Correlation matrix	Default unless COV is specified
Cov	Covariance matrix	Default if COV is specified

Table Name	Description	Option
Eigenvalues	Eigenvalues	
Eigenvectors	Eigenvectors	
NObsNVar	Number of observations, variables, and (partial) variables	
SimpleStatistics	Simple statistics	
TotalVariance	Total variance	COV
ODS Tables Created by the PARTIAL Statement		
ParCorr	Partial correlation matrix	
ParCov	Uncorrected partial covariance matrix	COV
RegCoef	Regression coefficients	COV
RSquareRMSE	Regression statistics: R-Squares and RMSEs	
StdRegCoef	Standardized regression coefficients	

Table 12.74 ODS Table Names Produced by the PRINQUAL Procedure

Table Name	Description	Option
ConvergenceStatus	Convergence status	
Footnotes	Iteration history footnotes	
ODS Tables Created by the PROC Statement		
MAC	MAC iteration history	METHOD=MAC
MGV	MGV iteration history	METHOD=MGV
MTV	MTV iteration history	METHOD=MTV

Table 12.75 ODS Table Names Produced by the PROBIT Procedure

Table Name	Description	Option
ODS Tables Created by the CLASS Statement		

Table Name	Description	Option
ClassLevels	Class variable levels	
	ODS Tables Created by the MODEL Statement	
ConvergenceStatus	Convergence status	
CorrB	Parameter estimate correlation matrix	CORRB
CovB	Parameter estimate covariance matrix	COVB
CovTolerance	Covariance matrix for location and scale	
GoodnessOfFit	Goodness of fit tests	LACKFIT
IterHistory	Iteration history	ITPRINT
LagrangeStatistics	Lagrange statistics	NOINT
LastGrad	Last evaluation of the gradient	ITPRINT
LastHess	Last evaluation of the Hessian	ITPRINT
LogProbitAnalysis	Probit analysis for log dose	INVERSECL
ModelInfo	Model information	
MuSigma	Location and scale	
ParameterEstimates	Parameter estimates	
ParmInfo	Parameter indices	
ProbitAnalysis	Probit analysis for linear dose	INVERSECL
ResponseLevels	Response-covariate profile	LACKFIT
ResponseProfiles	Counts for ordinal data	
Type3Analysis	Type 3 tests	

Table 12.76 ODS Table Names Produced by the REG Procedure

Table Name	Description	Option
	ODS Tables Created by the MODEL Statement	
ACovEst	Consistent covariance of estimates matrix	ALL or ACOV
ANOVA	Model ANOVA table	
CollinDiag	Collinearity diagnostics table	COLLIN
CollinDiagNoInt	Collinearity diagnostics for no intercept model	COLLINOINT

Table Name	Description	Option
ConditionBounds	Bounds on condition number	(SELECTION=BACKWARD, FORWARD, STEPWISE, MAXR, or MINR) and DETAILS
CorrB	Correlation of estimates	CORRB
CovB	Covariance of estimates	COVB
CrossProducts	Bordered model X"X matrix	ALL or XPX
DWStatistic	Durbin-Watson statistic	ALL or DW
DependenceEquations	Linear dependence equations	
EntryStatistics	Entry statistics for selection methods	(SELECTION=BACKWARD, FORWARD, STEPWISE, MAXR, or MINR) and DETAILS
FitStatistics	Model fit statistics	
InvXPX	Bordered X"X inverse matrix	I
OutputStatistics	Output statistics table	ALL, CLI, CLM, INFLUENCE, P, or R
ParameterEstimates	Model parameter estimates	
RemovalStatistics	Removal statistics for selection methods	(SELECTION=BACKWARD, STEPWISE, MAXR, or MINR) and DETAILS
ResidualStatistics	Residual statistics and PRESS statistic	ALL, CLI, CLM, INFLUENCE, P, or R
SelParmEst	Parameter estimates for selection methods	SELECTION=BACKWARD, FORWARD, STEPWISE, MAXR, or MINR
SelectionSummary	Selection summary for forward, backward, and stepwise methods	SELECTION=BACKWARD, FORWARD, or STEPWISE
SeqParmEst	Sequential parameter estimates	SEQB
SpecTest	White's heteroscedasticity test	ALL or SPEC
SubsetSelSummary	Selection summary for R-Square, adj-RSq, and Cp methods	SELECTION=RSQUARE, ADJRSQ, or CP
ODS Tables Created by the MTEST Statement		
CanCorr	Canonical correlations for hypothesis combinations	CANPRINT
Eigenvalues	MTest eigenvalues	CANPRINT
Eigenvectors	MTest eigenvectors	CANPRINT

Table Name	Description	Option
ErrorPlusHypothesis	MTest error plus hypothesis matrix H+E	PRINT
ErrorSSCP	MTest error matrix E	PRINT
HypothesisSSCP	MTest hypothesis matrix	PRINT
InvMTestCov	Inv(L Ginv(X"X)L") and Inv(Lb-c)	DETAILS
MTestCov	L Ginv(X"X) L" and Lb-c	DETAILS
MTransform	MTest matrix M, across dependents	DETAILS
MultStat	Multivariate test statistics	
	ODS Tables Created by the PROC Statement	
Corr	Correlation matrix for analysis variables	ALL or CORR
SimpleStatistics	Simple statistics for analysis variables	ALL or SIMPLE
USSCP	Uncorrected SSCP matrix for analysis variables	ALL or USSCP
	ODS Tables Created by the TEST Statement	
ACovTestANOVA	Test ANOVA using ACOV estimates	ACOV (MODEL statement)
InvTestCov	Inv(L Ginv(X"X)L") and Inv(Lb-c)	PRINT
TestANOVA	Test ANOVA table	
TestCov	L Ginv(X"X) L" and Lb-c	PRINT

Table 12.77 ODS Table Names Produced by the ROBUSTREG Procedure

Table Name	Description	Option
	ODS Tables Created by the CLASS Statement	
ClassLevels	Class variable levels	
	ODS Tables Created by the MODEL Statement	
CorrB	Parameter estimate correlation matrix	CORRB

Table Name	Description	Option
CovB	Parameter estimate covariance matrix	COVB
Diagnostics	Outlier diagnostics	DIAGNOSTICS
DiagSummary	Summary of the outlier diagnostics	
GoodFit	R2, deviance, AIC, and BIC	
ModelInfo	Model information	
ParameterEstimates	Parameter estimates	
ParmInfo	Parameter indices	
SummaryStatistics	Summary statistics for model variables	
ODS Tables Created by the PROC Statement		
BestEstimates	Best final estimates for LTS	SUBANALYSIS
BestSubEstimates	Best estimates for each subgroup	SUBANALYSIS
BiasTest	Bias test for MM estimation	BIASTEST
CStep	C-Step for LTS fitting	SUBANALYSIS
Groups	Groups for LTS fitting	SUBANALYSIS
InitLTSProfile	Profile for initial LTS estimate	METHOD
InitSProfile	Profile for initial S estimate	METHOD
LTSEstimates	LTS parameter estimates	METHOD
LTSLocationScale	Location and scale for LTS	METHOD
LTSProfile	Profile for LTS estimate	METHOD
LTSRsquare	R2 for LTS estimate	METHOD
MMProfile	Profile for MM estimate	METHOD
ParameterEstimatesF	Final weighted LS estimates	FWLS
SProfile	Profile for S estimate	METHOD
ODS Tables Created by the TEST Statement		
ParameterEstimatesR	Reduced parameter estimates	
TestsProfile	Results for tests	

Table 12.78 ODS Table Names Produced by the RSREG Procedure

Table Name	Description
Coding	Coding coefficients for the independent variables
ErrorANOVA	Error analysis of variance
FactorANOVA	Factor analysis of variance
FitStatistics	Overall statistics for fit
ModelANOVA	Model analysis of variance
ParameterEstimates	Estimated linear parameters
Spectral	Spectral analysis
StationaryPoint	Stationary point of response surface
ODS Tables Created by the RIDGE Statement	
Ridge	Ridge analysis for optimum response

Table 12.79 ODS Table Names Produced by the STDIZE Procedure

Table Name	Description	Option
Statistics	Location and scale measures	PSTAT

Table 12.80 ODS Table Names Produced by the STEPDISC Procedure

Table Name	Description	Option
BCorr	Between-class correlations	BCORR
BCov	Between-class covariances	BCOV
BSSCP	Between-class SSCP matrix	BSSCP
Counts	Number of observations, variables, classes, and DF	
CovDF	DF for covariance matrices, not printed	Any *COV option
Levels	Class level information	
Messages	Entry/removal messages	

Table Name	Description	Option
Multivariate	Multivariate statistics	
PCorr	Pooled within-class correlations	PCORR
PCov	Pooled within-class covariances	PCOV
PSSCP	Pooled within-class SSCP matrix	PSSCP
PStdMeans	Pooled standardized class means	STDMEAN
SimpleStatistics	Simple statistics	SIMPLE
Steps	Stepwise selection entry/ removal	
Summary	Stepwise selection summary	
TCorr	Total-sample correlations	TCORR
TCov	Total-sample covariances	TCOV
TSSCP	Total-sample SSCP matrix	TSSCP
TStdMeans	Total standardized class means	STDMEAN
Variables	Variable lists	
WCorr	Within-class correlations	WCORR
WCov	Within-class covariances	WCOV
WSSCP	Within-class SSCP matrices	WSSCP

Table 12.81 ODS Table Names Produced by the SURVEYFREQ Procedure

Table Name	Description	Statement	Option
ChiSq	Chi-square test	TABLES	CHISQ
ChiSq1	Modified chi-square test	TABLES	CHISQ1
CrossTabs	Crosstabulation table	TABLES	(n-way table request, n > 1)
LRChiSq	Likelihood ratio test	TABLES	LRCHISQ
LRChiSq1	Modified likelihood ratio test	TABLES	LRCHISQ1
OneWay	One-way frequency table	PROC or TABLES	(With no TABLES statement) (One-way table request)
StrataInfo	Stratum information	STRATA	LIST

Table Name	Description	Statement	Option
Summary	Data summary	PROC	Default
TableSummary	Table summary (not displayed)	TABLES	Default
WChiSq	Wald chi-square test	TABLES	WCHISQ
WLLChiSq	Wald log-linear chi-square test	TABLES	WLLCHISQ

Table 12.82 ODS Table Names Produced by the SURVEYLOGISTIC Procedure

Table Name	Description	Statement	Option
ClassLevelInfo	CLASS variable levels and design variables	MODEL	Default (with CLASS vars)
CLOdds	Wald's confidence limits for odds ratios	MODEL	CLODDS
CLparmWald	Wald's confidence limits for parameters	MODEL	CLPARM
ContrastCoeff	L matrix from CONTRAST	CONTRAST	E
ContrastEstimate	Estimates from CONTRAST	CONTRAST	ESTIMATE=
ContrastTest	Wald test for CONTRAST	CONTRAST	Default
ConvergenceStatus	Convergence status	MODEL	Default
CorrB	Estimated correlation matrix of parameter estimators	MODEL	CORRB
CovB	Estimated covariance matrix of parameter estimators	MODEL	CovB
CumulativeModelTest	Test of the cumulative model assumption	MODEL	(Ordinal response)
DesignSummary	Design summary	STRATA | CLUSTER	Default
FitStatistics	Model fit statistics	MODEL	Default
GlobalTests	Test for global null hypothesis	MODEL	Default
IterHistory	Iteration history	MODEL	ITPRINT
LastGradient	Last evaluation of gradient	MODEL	ITPRINT

Table Name	Description	Statement	Option
LogLikeChange	Final change in the log likelihood	MODEL	ITPRINT
ModelInfo	Model information	PROC	Default
NObs	Number of observations	PROC	Default
OddsRatios	Odds ratios	MODEL	Default
ParameterEstimates	Maximum likelihood estimates of model parameters	MODEL	Default
RSquare	R-square	MODEL	RSQUARE
ResponseProfile	Response profile	PROC	Default
SimpleStatistics	Summary statistics for explanatory variables	PROC	SIMPLE
StrataInfo	Stratum information	STRATA	LIST
TestPrint1	L[cov(b)] L" and Lb-c	TEST	PRINT
TestPrint2	Ginv(L[cov(b)]L") and Ginv(L[cov(b)]L")(Lb-c)	TEST	PRINT
TestStmts	Linear hypotheses testing results	TEST	Default
TypeIII	Type III tests of effects	MODEL	Default (with CLASS variables)

Table 12.83 ODS Table Names Produced by the SURVEYMEANS Procedure

Table Name	Description	Option
	ODS Tables Created by the CLASS Statement	
ClassVarInfo	Class level information	
	ODS Tables Created by the DOMAIN Statement	
Domain	Statistics in domains	
	ODS Tables Created by the PROC Statement	
Statistics	Statistics	
Summary	Data summary	
	ODS Tables Created by the RATIO Statement	

Table Name	Description	Option
Ratio	Statistics for ratios	
	ODS Tables Created by the STRATA Statement	
StrataInfo	Stratum information	LIST

Table 12.84 ODS Table Names Produced by the SURVEYREG Procedure

Table Name	Description	Option
	ODS Tables Created by the CLASS Statement	
ClassVarInfo	Class level information	
	ODS Tables Created by the CLUSTER Statement	
DesignSummary	Design summary	
	ODS Tables Created by the CONTRAST Statement	
ContrastCoef	Coefficients of contrast	E
Contrasts	Analysis of contrasts	
	ODS Tables Created by the ESTIMATE Statement	
EstimateCoef	Coefficients of estimate	E
Estimates	Analysis of estimable functions	
	ODS Tables Created by the MODEL Statement	
ANOVA	ANOVA for dependent variable	ANOVA
CovB	Covariance of estimated regression coefficients	COVB
DataSummary	Data summary	
Effects	Tests of model effects	
FitStatistics	Fit statistics	
InvXPX	Inverse matrix of X"X	INV
ParameterEstimates	Estimated regression coefficients	
XPX	X"X matrix	XPX

Table Name	Description	Option
ODS Tables Created by the STRATA Statement		
DesignSummary	Data summary	
StrataInfo	Stratum information	LIST

Table 12.85 ODS Table Names Produced by the SURVEYSELECT Procedure

Table Name	Description
ODS Tables Created by the PROC Statement	
Method	Sample selection method
Summary	Sample selection summary

Table 12.86 ODS Table Names Produced by the TPSPLINE Procedure

Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
GCVFunction	GCV table	LOGNLAMBDA or LAMBDA
ODS Tables Created by the PROC Statement		
DataSummary	Data summary	
FitStatistics	Model fit statistics	
FitSummary	Fit parameters and fit summary	

Table 12.87 ODS Table Names Produced by the TRANSREG Procedure

Table Name	Description	Option
ConvergenceStatus	Convergence status	
Equation	Linear dependency equation	Less-than-full-rank model
Footnotes	Iteration history footnotes	

Table Name	Description	Option
	ODS Tables Created by the MODEL Statement	
BoxCox	Box-Cox transformation results	BOXCOX
SplineCoef	Spline coefficients	SPLINE or MSPLINE
	ODS Tables Created by the MODEL and PROC Statements	
NObs	ANOVA	TEST or SS2
ClassLevels	ANOVA	TEST or SS2
ANOVA	ANOVA	TEST or SS2
LiberalANOVA	ANOVA	TEST or SS2
ConservANOVA	ANOVA	TEST or SS2
FitStatistics	Fit statistics like R-Square	TEST or SS2
LiberalFitStatistics	Fit statistics	TEST or SS2
ConservFitStatistics	Fit statistics	TEST or SS2
MVANOVA	Multivariate ANOVA	TEST or SS2
LiberalMVANOVA	Multivariate ANOVA	TEST or SS2
ConservANOVA	Multivariate ANOVA	TEST or SS2
Coef	Regression results	SS2
LiberalCoef	Regression results	SS2
ConservCoef	Regression results	SS2
MVCoef	Multivariate regression results	SS2
LiberalMVCoef	Multivariate regression results	SS2
ConservMVCoef	Multivariate regression results	SS2
Utilities	Conjoint analysis utilities	UTILITY
LiberalUtilities	Conjoint analysis utilities	UTILITY
ConservUtilities	Conjoint analysis utilities	UTILITY
Details	Model details	DETAIL
Univariate	Univariate iteration history	METHOD=UNIVARIATE
MORALS	MORALS iteration history	METHOD=MORALS
CANALS	CANALS iteration history	METHOD=CANALS
Redundancy	Redundancy iteration history	METHOD=REDUNDANCY
TestIterations	Hypothesis test iterations iteration history	SS2

Table 12.88 ODS Table Names Produced by the TREE Procedure

Table Name	Description	Option
	ODS Tables Created by the PROC Statement	
Tree	Line-printer plot of the tree	LINEPRINTER
TreeListing	Line-printer listing of all nodes in the tree	LIST

Table 12.89 ODS Table Names Produced by the TTEST Procedure

Table Name	Description
Statistics	Univariate summary statistics
TTests	*t*-tests
	ODS Tables Created by the CLASS Statement
Equality	Tests for equality of variance

Table 12.90 ODS Table Names Produced by the VARCLUS Procedure

Table Name	Description	Option
ClusterQuality	Cluster quality	
ClusterStructure	Cluster structure	
ClusterSummary	Cluster summary	
ConvergenceStatus	Convergence status	
Corr	Correlations	CORR
DataOptSummary	Data and options summary table	
InterClusterCorr	Inter-cluster correlations	
IterHistory	Iteration history	TRACE
RSquare	Cluster R-Square	
SimpleStatistics	Simple statistics	SIMPLE
StdScoreCoef	Standardized scoring coefficients	

Table 12.91 ODS Table Names Produced by the VARCOMP Procedure

Table Name	Description	Option
ClassLevels	Class level information	
ConvergenceStatus	Convergence status	
Estimates	Variance component estimates (one variable)	
Estimates*n*	Variance component estimates (multiple variables)	
NObs	Number of observations	
	ODS Tables Created by the METHOD Statement	
ANOVA	Type 1 analysis of variance	TYPE1
AsyCov	Asymptotic covariance matrix of estimates	ML or REML
DepVar	Dependent variable (one variable)	TYPE1, REML, or ML
DepVar*n*	Dependent variable *n* (multiple variables)	TYPE1, REML, or ML
DependentInfo	Dependent variable information (multiple variables)	MIVQUE0
IterHistory	Iteration history	ML or REML
SCCP	Sum of squares matrix (one variable)	MIVQUE0
SCCP*n*	Sum of squares matrix (multiple variable)	MIVQUE0

ODS Table Names and the SAS/ETS Procedures That Produce Them

This table lists the output object table names which SAS/ETS procedures produce. You must license SAS/ETS software in order to produce these output objects. The table provides the name of each table, a description of what the table contains, and the option, if any, that creates the output object table. For more information about SAS/ETS procedures, see *SAS/ETS User's Guide*.

Table 12.92 ODS Table Names Produced by the ARIMA Procedure

Table Name	Description	Option
	ODS Tables Created by the IDENTIFY Statement	
DescStats	Descriptive statistics	
InputDescStats	Input descriptive statistics	
CorrGraph	Correlations graph	
StationarityTest	Stationarity tests	STATIONARITY
TentativeOrders	Tentative order selections	MINIC, ESACF, or SCAN
PACFGraph	Partial autocorrelations graph	
IACFGraph	Inverse autocorrelations graph	
ChiSqAuto	Chi-Square statistics table for autocorrelation	
ChiSqCross	Chi-Square statistics table for cross-correlations	CROSSCORR=
MINIC	Minimum information criterion	MINIC
ESACF	Extended sample autocorrelation function	ESACF
ESACFPValues	ESACF probability values	ESACF
SCAN	Squared canonical correlation estimates	SCAN
SCANValues	SCAN Chi-Square[1] probability values	
	ODS Tables Created by the ESTIMATE Statement	
FitStatistics	Fit statistics	
ARPolynomial	Filter equations	
MAPolynomial	Filter equations	
NumPolynomial	Filter equations	
DenPolynomial	Filter equations	
ParameterEstimates	Parameter estimates	
ChiSqAuto	Chi-Square statistics table for autocorrelation	
ChiSqCross	Chi-Square statistics table for cross-correlations	
InitialAREstimates	Initial autoregressive parameter estimates	
InitialMAEstimates	Initial moving average parameter estimates	

Table Name	Description	Option
PrelimEstimates	Preliminary estimation	
IterHistory	Conditional least squares estimation	METHOD=CLS
OptSummary	ARIMA estimation optimization	PRINTALL
ModelDescription	Model description	
InputDescription	Input description	
ObjectiveGrid	Objective function grid matrix	GRID
CorrB	Correlations of the estimates	
ODS Tables Created by the OUTLIER Statement		
OutlierDetails	Detected outliers	
ODS Tables Created by the FORECAST Statement		
Forecasts	Fit statistics	

Table 12.93 ODS Table Names Produced by the AUTOREG Procedure

Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
FitSummary	Summary of regression	
SummaryDepVarCen	Summary of regression (centered dependent variable)	CENTER
SummaryNoIntercept	Summary of regression (no intercept)	NOINT
YWIterSSE	Yule-Walker iteration sum of squared error	METHOD=ITYW
PreMSE	Preliminary MSEs	NLAG=
Dependent	Dependent variable	
DependenceEquations	Linear dependence equation	
ARCHTest	Q and LM tests for ARCH disturbances	ARCHTEST
ChowTest	Chow test and predictive chow test	CHOW= or PCHOW=
Godfrey	Godfrey's serial correlation test	GODFREY or GODFREY=

Table Name	Description	Option
PhilPerron	Phillips-Perron unit root test	STATIONARITY=, (PHILLIPS<=()>), (no regressor)
PhilOul	Phillips-Ouliaris cointegration test	STATIONARITY=, (PHILLIPS<=()>), (has regressor)
ResetTest	Ramsey's RESET test	RESET
ARParameterEstimates	Estimates of autoregressive parameters	NLAG=
CorrGraph	Estimates of autocorrelations	NLAG=
BackStep	Backward elimination of autoregressive terms	BACKSTEP
ExpAutocorr	Expected autocorrelations	NLAG=
IterHistory	Iteration history	ITPRINT
ParameterEstimates	Parameter estimates	
ParameterEstimatesGivenAR	Parameter estimates assuming AR parameters are given	NLAG=
PartialAutoCorr	Partial autocorrelation	PARTIAL
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
CholeskyFactor	Cholesky root of gamma	ALL
Coefficients	Coefficients for first NLAG observations	COEF
GammaInverse	Gamma inverse	GINV
ConvergenceStatus	Convergence status table	
DWTest	Durbin-Watson statistics	DW=
ODS Tables Created by the RESTRICT Statement		
Restrict	Restriction table	
ODS Tables Created by the TEST Statement		
FTest	F test	
WaldTest	Wald test	TYPE=WALD

Table 12.94 ODS Table Names Produced by the ENTROPY Procedure

Table Name	Description
ConvCrit	Convergence criteria for estimation
ConvergenceStatus	Convergence status
DatasetOptions	Data sets used
MinSummary	Number of parameters, estimation kind
ObsUsed	Observations read, used, and missing
ParameterEstimates	Parameter estimates
ResidSummary	Summary of the SSE, MSE for the equations
TestResults	Test statement table

Table 12.95 ODS Table Names Produced by the LOAN Procedure

Table Name	Description	Option
ODS Tables Created by the PROC LOAN, FIXED, ARM, BALLOON, and BUYDOWN Statements		
Repayment	Loan repayment schedule	SCHEDULE
ODS Tables Created by the FIXED, ARM, BALLOON, and BUYDOWN Statements		
LoanSummary	Loan summary	
RateList	Rates and payments	
PrepayList	Prepayments and periods	PREPAYMENTS=
ODS Tables Created by the BALLOON Statement		
BalloonList	Balloon payments and periods	
ODS Tables Created by the COMPARE Statement		
Comparison	Loan comparison report	

Table 12.96 ODS Table Names Produced by the MDC Procedure

Table Name	Description	Option
	ODS Tables Created by the MODEL Statement	
FitSummary	Summary of nonlinear estimation	
ResponseProfile	Response profile	
GoodnessOfFit	Pseudo-R^2 measures	
ParameterEstimates	Parameter estimates	
LinConSol	Linearly independent active linear constraints	
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB

Table 12.97 ODS Table Names Produced by the MODEL Procedure

Table Name	Description	Option
	ODS Tables Created by the FIT Statement	
AugGMMCovariance	Cross products matrix	GMM
ChowTest	Structural change test	CHOW=
CollinDiagnostics	Collinearity diagnostics	
ConfInterval	Profile likelihood confidence intervals	PRL=
ConvCrit	Convergence criteria for estimation	
ConvergenceStatus	Convergence status	
CorrB	Correlations of parameters	COVB or CORRB
CorrResiduals	Correlations of residuals	CORRS or COVS
CovB	Covariance of parameters	COVB or CORRB
CovResiduals	Covariance of residuals	CORRS or COVS
Crossproducts	Cross products matrix	ITALL or ITPRINT
DatasetOptions	Data sets used	
DetResidCov	Determinant of the residuals	DETAILS
DWTest	Durbin-Watson test	DW=

Table Name	Description	Option
Equations	List of equations to estimate	
EstSummaryMiss	Model summary statistics for PAIRWISE	MISSING=
EstSummaryStats	Objective, objective * N	
GMMCovariance	Cross products matrix	GMM
Godfrey	Godfrey's serial correlation test	GF=
HausmanTest	Hausman's test table	HAUSMAN
HeteroTest	Heteroscedasticity test tables	BREUSCH or PAGEN
InvXPXMat	X"X inverse for system	I
IterInfo	Iteration printing	ITALL or ITPRINT
LagLength	Model lag length	
MinSummary	Number of parameters, estimation kind	
MissingValues	Missing values generated by the program	
ModSummary	List of all categorized values	
ModVars	List of model variables and parameters	
NormalityTest	Normality test table	NORMAL
ObsSummary	Identifies observations with errors	
ObsUsed	Observations read, used, and missing	Default
ParameterEstimates	Parameter estimates	
ParmChange	Parameter change vector	
ResidSummary	Summary of the SSE, MSE for the equations	
SizeInfo	Storage requirement for estimation	DETAILS
TermEstimates	Nonlinear OLS and ITOLS estimates	OLS or ITOLS
TestResults	Test statement table	
WgtVar	The name of the weight variable	
XPXMat	X"X for system	XPX
	ODS Tables Created by the SOLVE Statement	
DatasetOptions	Data sets used	
DescriptiveStatistics	Descriptive statistics	STATS

Table Name	Description	Option
FitStatistics	Fit statistics for simulation	STATS
LagLength	Model lag length	
ModSummary	List of all categorized variables	
ObsSummary	Simulation trace output	SOLVEPRINT
ObsUsed	Observations resa, used, and missing	
SimulationSummary	Number of variables solved for	
SolutionVarList	Solution variable lists	
TheilRelStats	Theil relative change error statistics	THEIL
TheilStats	Theil forecast error statistics	THEIL
ODS Tables Created by the FIT and SOLVE Statements		
AdjacencyMatrix	Adjacency graph	GRAPH
BlockAnalysis	Block analysis	BLOCK
CodeDependency	Variable cross reference	LISTDEP
CodeList	List of programs statements	LISTCODE
CrossReference	Cross reference listing for program	
DepStructure	Dependency structure for the system	BLOCK
DerList	Derivative variables	LISTDER
InterIntg	Integration iteration output	INTGPRINT
MemUsage	Memory usage statistics	MEMORYUSE
ParmReadIn	Parameter estimates read in	ESTDATA=
ProgList	List of compiled program data	
RangeInfo	RANGE statement specification	
SortAdjacencyMatrix	Sorted adjacency graph	GRAPH
TransitiveClosure	Transitive closure graph	GRAPH

Table 12.98 ODS Table Names Produced by the PDLREG Procedure

Table Name	Description	Option
ODS Tables Created by the MODEL Statement		

Table Name	Description	Option
ARParameterEstimates	Estimates of autoregressive parameters	NLAG=
CholeskyFactor	Cholesky root of gamma	
Coefficients	Coefficients for first NLAG observations	NLAG=
ConvergenceStatus	Convergence status table	
CorrB	Correlation of parameter estimates	CORRB
CorrGraph	Estimates of autocorrelations	NLAG=
CovB	Covariance of parameter estimates	COVB
DependenceEquations	Linear dependence equation	
Dependent	Dependent variable	
DWTest	Durbin-Watson statistics	DW=
ExpAutocorr	Expected autocorrelations	NLAG=
FitSummary	Summary of regression	
GammaInverse	Gamma inverse	
IterHistory	Iteration history	ITPRINT
LagDist	Lag distribution	ALL
ParameterEstimates	Parameter estimates	
ParameterEstimatesGivenAR	Parameter estimates assuming AR parameters are given	NLAG=
PartialAutoCorr	Partial autocorrelation	PARTIAL
PreMSE	Preliminary MSE	NLAG=
XPXIMatrix	Inverse X"X matrix	XPX
XPXMatrix	X"X matrix	XPX
YWIterSSE	Yule-Walker iteration sum of squared error	METHOD=ITYW

ODS Tables Created by the RESTRICT Statement

Restrict	Restriction table

Table 12.99 ODS Table Names Produced by the SIMLIN Procedure

Table Name	Description	Option
Endogenous	Structural coefficients for endogenous variables	
LaggedEndogenous	Structural coefficients for lagged endogenous variables	
Exogenous Structural	Coefficients for exogenous variables	
InverseCoeff	Inverse coefficient matrix for endogenous variables	
RedFormLagEndo	Reduced form for lagged endogenous variables	
RedFormExog	Reduced form for exogenous variables	
InterimMult	Interim multipliers	INTERIM=option
TotalMult	Total multipliers	TOTAL=option
FitStatistics	Fit statistics	

Table 12.100 ODS Table Names Produced by the SPECTRA Procedure

Table Name	Description	Option
WhiteNoiseTest	White noise test	WHITETEST
Kappa	Fishers kappa	WHITETEST
Bartlett	Bartletts Kolmogorov-Smirnov statistic	WHITETEST

Table 12.101 ODS Table Names Produced by the STATESPACE Procedure

Table Name	Description	Option
NObs	Number of observations	
Summary	Simple summary statistics table	
InfoCriterion	Information criterion table	

Table Name	Description	Option
CovLags	Covariance matrices of input series	PRINTOUT=LONG
CorrLags	Correlation matrices of input series	PRINTOUT=LONG
PartialAR	Partial autoregressive matrices	PRINTOUT=LONG
YWEstimates	Yule-Walker estimates for minimum AIC	
CovResiduals	Covariance of residuals	PRINTOUT=LONG
CorrResiduals	Residual correlations from AR models	PRINTOUT=LONG
StateVector	State vector table	
CorrGraph	Schematic representation of correlations	
TransitionMatrix	Transition matrix	
InputMatrix	Input matrix	
VarInnov	Variance matrix for the innovation	
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	COVB
CanCorr	Canonical correlation analysis	CANCORR
IterHistory	Iterative fitting table	ITPRINT
ParameterEstimates	Parameter estimates table	
Forecasts	Forecasts table	PRINT
ConvergenceStatus	Convergence status table	

Table 12.102 ODS Table Names Produced by the SYSLIN Procedure

Table Name	Description	Option
ANOVA	Summary of the SSE, MSE for the equations	
AugXPXMat	Model crossproducts	XPX
AutoCorrStat	Autocorrelation statistics	
ConvCrit	Convergence criteria for estimation	
ConvergenceStatus	Convergence status	

Table Name	Description	Option
CorrB	Correlations of parameters	CORRB
CorrResiduals	Correlations of residuals	CORRS
CovB	Covariance of parameters	COVB
CovResiduals	Covariance of residuals	COVS
Endomat	Endogenous variables	
Equations	List of equations to estimates	
ExogMat	Exogenous variables	
FitStatistics	Statistics of fit	
InvCorrResiduals	Inverse correlations of residuals	CORRS
InvCovResiduals	Inverse covariance of residuals	COVS
InvEndoMat	Inverse endogenous variables	
InvXPX	X"X inverse for system	I
IterHistory	Iteration printing	ITALL or ITPRINT
MissingValues	Missing values generated by the program	
ModelVars	Name and label for the model	
ParameterEstimates	Parameter estimates	
RedMat	Reduced form	REDUCED
SimpleStatistics	Descriptive statistics	SIMPLE
SSCP	Model crossproducts	
TestResults	Test for overidentifying restrictions	
Weight	Weighted model statistics	
YPY	Y"Y matrices	USSCP2

Table 12.103 ODS Table Names Produced by the TSCSREG Procedure

Table Name	Description	Option
	ODS Tables Created by the MODEL Statement	
ModelDescription	Model description	
FitStatistics	Fit statistics	
FixedEffectsTest	F test for no fixed tests	
ParameterEstimates	Parameter estimates	
CovB	Covariance of parameter estimates	

Table Name	Description	Option
CorrB	Correlations of parameter estimates	
VarianceComponents	Variance component estimates	
RandomEffectsTest	Hausman test for random effects	
AR1Estimates	First order autoregressive parameter estimates	
EstimatedPhiMatrix	Estimated phi matrix	PARKS
EstimatedAutocovariances	Estimates of autocovariances	PARKS
ODS Tables Created by the TEST Statement		
TestResults	Test results	

Table 12.104 ODS Table Names Produced by the TIMESERIES Procedure

Table Name	Description
ODS Tables Created by the PRINT=DECOMP Option	
SeasonalDecomposition	Seasonal decomposition
ODS Tables Created by the PRINT=DESCSTATS Option	
DescStats	Descriptive statistics
ODS Tables Created by the PRINT=SEASONS Option	
SeasonStatistics	Season statistics
ODS Tables Created by the PRINT=TRENDS Option	
TrendStatistics	Trend statistics

Table 12.105 ODS Table Names Produced by the VARMAX Procedure

Table Name	Description	Option
ODS Tables Created by the MODEL Statement		

Table Name	Description	Option
AccumImpulse	Accumulated impulse response matrices	IMPULSE=(ACCUM) or IMPULSE=(ALL)
AccumImpulsX	Accumulated transfer function matrices	IMPULSX=(ACCUM) or IMPULSX=(ALL)
Alpha	α coefficients	JOHANSEN=
AlphaInECM	α coefficients	ECM=
AlphaOnDrift	α coefficients on restriction of a deterministic term	JOHANSEN=
AlphaBetaInECM	$\pi=\alpha\beta'$ coefficients	ECM=
ArchCoef	ARCH coefficients	GARCH=
ARCoef	AR coefficients	P= or DYNAMIC with P=
ARRoots	Roots of AR characteristic polynomial	ROOTS
Beta	β coefficients	JOHANSEN=
BetaInECM	β coefficients	ECM=
BetaOnDrift	β coefficients on restriction of a deterministic term	JOHANSEN=
Constant	Constant estimates	Without NOINT
CorrB	Correlations of parameter estimates	CORRB
CorrResiduals	Cross-correlations of residuals	
CorrResidualsGraph	Schematic representation of residual cross-correlations	
CorrGraph	Schematic representation of sample cross-correlations	CORRX or CORRY
CorrXLags	Cross-correlation matrices of independent series	CORRX
CorrYLags	Cross-correlation matrices of dependent series	CORRY
CovB	Covariance of parameter estimates	COVB
CovInnov	Covariance matrix for the innovation	
CovPredError	Covariance matrices of the prediction error	COVPE
CovResiduals	Cross-covariance matrices of residuals	
CovXLags	Cross-covariance matrices of independent series	COVX
CovYLags	Cross-correlations matrices of dependent series	COVY

Table Name	Description	Option
DecompCovPredError	Decomposition of the prediction error covariance	DECOMPOSE
DFTest	Dickey-Fuller tests	DFTEST
DriftHypo	Hypothesis of different deterministic terms in cointegration rank test	JOHANSEN=
DrifyHypoTest	Test hypothesis of different deterministic terms in cointegration rank test	JOHANSEN=
EigenvalueI2	Eigenvalues in integrated order 2	JOHANSEN= (IORDER=2)
Eta	η coefficients	JOHANSEN= (IORDER=2)
GARCHParameterEstimates	GARCH parameter estimates table	GARCH=
GARCHParameterGraph	Schematic representation of the garch parameters	
GARCHRoots	Roots of GARCH characteristic polynomial	GARCH=
GARCHCoef	GARCH coefficients	GARCH=
GARCHConstant	GARCH constant estimates	GARCH=
InfiniteARRepresent	Infinite order AR representation	IARR
InfoCriterion	Information criterion	
LinearTrend	Linear trend estimates	TREND=
MACoef	MA coefficients	Q=
MARoots	Roots of MA characteristic polynomial	Q=
MaxTest	Cointegration rank test using the maximum eigenvalue	JOHANSEN= (TYPE=MAX)
MaxTestOnDrift	Cointegration rank test using the maximum eigenvalue on restriction of a deterministic term	JOHANSEN= (TYPE=MAX)
ModelType	Type of model	
NObs	Number of observations	
OrthoImpulse	Orthogonalized impulse response matrices	IMPULSE=(ORTH) or IMPULSE=(ALL)
ParameterEstimates	Parameter estimates table	
ParameterGraph	Schematic representation of the parameters	
PartialAR	Partial autoregression matrices	PARCOEF

Table Name	Description	Option
PartialARGraph	Schematic representation of partial autoregression	PARCOEF
PartialCanCorr	Partial canonical correlation analysis	PCANCORR
PartialCorr	Partial cross-correlation matrices	PCORR
PartialCorrGraph	Schematic representation of partial cross correlations	PCORR
PortmanteauTest	Chi-Square test table for residual cross-correlations	
ProportionDecomp	Proportions of prediction error covariance decomposition	DECOMPOSE
RankTestI2	Cointegration rank test in integrated order 2	JOHANSEN= (IORDER=2)
QuadTrend	Quadratic trend estimates	TREND=QUAD
SConstant	Seasonal constant estimates	NSEASON=
SimpleImpulse	Impulse response matrices	IMPULSE, IMPULSE=SIMPLE, or IMPULSE=(ALL)
SimpleImpulsX	Impulse response matrices in transfer function	IMPULSX, IMPULSX=(SIMPLE), or IMPULSX=(ALL)
Summary	Simple summary statistics	
SWTest	Common trends test	SW or SW=
TentativeOrders	Tentative order selection	MINIC or MINIC=
TraceTest	Cointegration rank test using the trace	JOHANSEN= (TYPE=TRACE)
TraceTestOnDrift	Cointegration rank test using the trace on restriction of a deterministic term	JOHANSEN= (TYPE=TRACE)
UnivarDiagnostAR	Check the AR disturbance for the residuals	
UnivarDiagnostCheck	Univariate model diagnostic checks	
UnivarDiagnostTest	Check the ARCH disturbance and normality for the residuals	
Xi	ξ coefficient matrix	JOHANSEN= (IORDER=2)
XLagCoef	Dependent coefficients	XLAG=
YWEstimates	Yule-Walker estimates	YW
ByVariable	Prints by variable	PRINTFORM=

Table Name	Description	Option
	ODS Tables Created by the COINTEG Statement	
AlphaInECM	α coefficients	
AlphaBetaInECM	$\pi = \alpha\beta'$ coefficients	
BetaInECM	β coefficients	
AlphaOnTest	α coefficients under restriction	H= or J=
BetaOnTest	β coefficients under restriction	H= or J=
RestrictMatrix	Restriction matrix for α or β	H= or J=
RestrictTest	Hypothesis testing of α or β	H= or J=
WeakExogeneity	Testing weak exogeneity of each dependent variable with respect to beta	EXOGENEITY
	ODS Tables Created by the CASUAL Statement	
Causality	Granger-Causality test	
	ODS Tables Created by the RESTRICT Statement	
Restrict	Restriction table	
	ODS Tables Created by the TEST Statement	
Test	Wald test	
	ODS Tables Created by the OUTPUT Statement	
Forecasts	Forecasts table	Without NOPRINT

Table 12.106 ODS Table Names Produced by the X11 Procedure

Table Name	Description	Option
	ODS Tables Created by the MONTHLY and QUARTERLY Statements	
Preface	X11 seasonal adjustment program information giving credits, dates, etc.	Always printed unless NOPRINT
A1	OriginalSeries	
A2	Prior monthly	

Table Name	Description	Option
A3	Original series adjusted for prior monthly factors	
A4	Prior trading day adjustment factors with and without length of month adjustments	
A5	Original series adjusted for priors	
B1	Original series or original series adjusted for priors	
B2	Trend cycle — centered nn-term moving average	
B3	Unmodified SI ratios	
B4	Replacement values for extreme SI ratios	
B5	Seasonal factors	
B6	Seasonally adjusted series	
B7	Trend cycle — Henderson curve	
B8	Unmodified SI ratios	
B9	Replacement values for extreme SI ratios	
B10	Seasonal factors	
B11	Seasonally adjusted series	
B13	Irregular series	
B15	Preliminary trading day regression	
B16	Trading day adjustment factors derived from regression	
B17	Preliminary weights for irregular components	
B18	Trading day adjustment factors from combined weights	
B19	Original series adjusted for preliminary combined TD weights	
C1	Original series adjusted for preliminary weights	
C2	Trend cycle — centered nn-term moving average	
C4	Modified SI ratios	
C5	Seasonal factors	
C6	Seasonally adjusted factors	

Table Name	Description	Option
C7	Trend cycle — Henderson curve	
C9	Modified CI ratios	
C10	Seasonal factors	
C11	Seasonally adjusted series	
C13	Irregular series	
C15	Final trading day regression	
C16	Trading day adjustment factors derived from regression	
C17	Final weights for irregular component	
C18	Trading day adjustment factors from combined weights	
C19	Original series adjusted for final combined TD weights	
D1	Original series adjusted for final weights on nn-term moving average	
D4	Modified SI ratios	
D5	Seasonal factors	
D6	Seasonally adjusted series	
D7	Trend cycle — Henderson curve	
D8	Final unmodified SI ratios	
D10	Final season factors	
D11	Final seasonally adjusted series	
D12	Final trend cycle — Henderson curve	
D13	Final irregular series	
E1	Original series modified for extremes	
E2	Modified seasonally adjusted series	
E3	Modified irregular series	
E5	Month-to-month changes in original series	
E6	Month-to-month changes in final seasonally adjusted series	
F1	MCD moving average	
A13	ARIMA forecasts	ARIMA statement
A14	ARIMA backcasts	ARIMA statement

Table Name	Description	Option
A15	ARIMA extrapolation	ARIMA statement
B14	Irregular values excluded from trading day regression	
C14	Irregular values excluded from trading day regression	
D9	Final replacement values	
PriorDailyWgts	Adjusted prior daily weights	
TDR_0	Final/preliminary trading day regression, part 1	MONTHLY only, TDREGR=ADJUST, TEST
TDR_1	Final/preliminary trading day regression, part 2	MONTHLY only, TDREGR=ADJUST, TEST
StandErrors	Standard errors of trading day adjustment factors	MONTHLY only, TDREGR=ADJUST, TEST
D9A	Year-to-year change in irregular and seasonal components and moving seasonality ratio	
StableSeasTest	Stable seasonality test	MONTHLY only
StableSeasFTest	Stable seasonality test	MONTHLY only
f2a	F2 summary measures, part 1	
f2b	F2 summary measures, part 2	
f2c	F2 summary measures, part 3	
f2d	I/C ratio for monthly/quarterly span	
f2f	Average percent change with regard to sign and standard over span	
E4	Differences or ratios of annual totals, original and adjusted series	
ChartG1	Chart G1	
ChartG2	Chart G2	

ODS Tables Created by the ARIMA Statement

CriteriaSummary	Criteria summary	ARIMA statement
ConvergeSummary	Convergence summary	
ArimaEst	ARIMA estimation results, part 1	
ArimaEst2	ARIMA estimation results, part 2	
Model_Summary	Model summary	

Table Name	Description	Option
Ljung_BoxQ	Table of Ljung-Box Q statistics	
A13	ARIMA forecasts	
A14	ARIMA backcasts	
A15	ARIMA extrapolation	
	ODS Tables Created by the SSPAN Statement	
SPR0A_1	S 0.A sliding spans analysis, number, and length of spans	
SpanDates	S 0.A sliding spans analysis: dates of spans	
SPR0B	S 0.B summary of F-tests for stable and moving seasonality	
SPR1_1	S 1.A range analysis of seasonal factors	
SPR1_b	S 1.B summary of range measures	
SPRXA	2XA.1 breakdown of differences by month or quarter	
SPRXB_2	S X.B histogram of flagged observation	
SPRXA_2	S X.A.2 breakdowns of differences by year	
MpdStats	S X.C: Statistics for maximum percentage differences	
S_X_A_3	S 2.X.3 breakdown summary of flagged observation	
SPR7_X	S 7.X sliding spans analysis	PRINTALL

Table 12.107 ODS Table Names Produced by the X12 Procedure

Table Name	Description
A1	Original series
A2	Prior-adjustment factors
RegParameterEstimates	Regression model parameter estimates
ACF	Autocorrelation factors
PACF	Partial autorrelation factors

Table Name	Description
ARMAIterationTolerances	Exact ARMA likelihood estimation iteration tolerances
IterHistory	ARMA iteration history
ARMAIterationSummary	Exact ARMA likelihood estimation iteration summary
RegressorGroupChiSq	Chi-Squared tests for groups of regressors
ARMAParameterEstimates	Exact ARMA maximum likelihood estimation
AvgFcstErr	Average absolute percentage error in within(out) sample fore(back)casts
Roots	(Non)seasonal (AR)MA roots
MLESummary	Estimation summary
ForecastCL	Forecasts, standard errors, and confidence limits
MV1	Original series adjusted for missing value regressors
A6	RegARIMA trading day component
A8	RegARIMA combined outlier component
A8AO	RegARIMA AO outlier component
A8LS	RegARIMA level change outlier component
A8TC	RegARIMA temporary change outlier component
B1	Prior adjusted or original series
C17	Final weight for irregular components
C20	Final extreme value adjusted factors
D1	Modified original data, D iteration
D7	Preliminary trend cycle, D iteration
D8	Final unmodified S-I ratios
D8A	Seasonality tests
D9	Final replacement values for extreme S-I ratios
D9A	Moving seasonality ratio
D10	Final seasonal factors
D10D	Final seasonal difference
D11	Final seasonally adjusted series
D12	Final trend cycle
D13	Final irregular series
D16	Combined adjustment factors
D16B	Final adjustment differences
D18	Combined calendar adjustment factors
E4	Ratios of annual totals

Table Name	Description
E5	Percent changes in original series
E6	Percent changes in final seasonally adjusted series
E7	Differences in final trend cycle
F2A-I	Summary measures
F3	Quality assessment statistics
F4	Day of the week trading day component factors
G	Spectral analysis

Concepts: Tabular Output and the TEMPLATE Procedure

Viewing the Contents of a Table Template

To view the contents of a table template, use the SAS windowing environment, the command line, or the TEMPLATE procedure.

- Using the SAS Windowing Environment
  1. From the menu, select **View ▶ Results.**
  2. In the Results window, select the **Results** folder. Right-click and select **Templates** to open the Templates window.
  3. Double-click Sashelp.Tmplmst to view the contents of that item store or directory.
  4. Double-click a directory to view the list of subdirectories and table templates that you wish to view. For example, the Base SAS table template Summary is the default template store for the summary tables created in the MEANS and SUMMARY procedures. Double-click the **Base** directory, and then double-click the Summary table.
- Using the Command Line
  1. To view the Templates window, submit this command:

     ```
     odstemplates
     ```

     The Templates window contains the item stores **Sasuser.Templat** and **Sashelp.Tmplmst**.
  2. When you double-click an item store, such as **Sashelp.Tmplmst**, that item store expands to list the directories where ODS templates are stored. The templates that SAS provides are in the item store Sashelp.Tmplmst.
  3. To view the table templates that SAS provides, double-click the item store that contains a table template, such as **Base**.
  4. Right-click the table template, such as **Summary**, and select **Open**. The table template is displayed in the Template Browser window.
- Using the TEMPLATE Procedure

1 The SOURCE statement writes the source code for the specified template to the SAS log. For example, if to view the source code for all the objects in Base SAS, submit this code.

```
proc template;
source base;
run;
```

Values in Table Columns and How They Are Justified

The process of justifying the values in columns in a listing output is determined by the format of the variable and the values of two attributes: JUST= and JUSTIFY=. It is a three-step process:

1 ODS puts the value into the format for the column. Character variables are left-justified within their format fields; numeric variables are right-justified.
2 ODS justifies the entire format field within the column width according to the value of the JUST= attribute for the column, or, if that attribute is not set, JUST= for the table. For example, if you right-justify the column, the format field is placed as far to the right as possible. However, the placement of the individual numbers and characters within the field does not change. Thus, decimal points remain aligned. If the column and the format field have the same width, then JUST= has no apparent effect because the format field occupies the entire column.
3 If you specify JUSTIFY=ON for the column or the table, ODS justifies the values within the column without regard to the format field. By default, JUSTIFY=OFF.

For example, consider this set of values:

```
123.45
234.5
.
987.654
```

If the values are formatted with a 6.2 format and displayed in a column with a width of 6, they appear this way, regardless of the value of JUST= (asterisks indicate the width of the column):

```
******
123.45
234.50
   .
987.65
```

If the width of the column increases to 8, then the value of JUST= does affect the placement of the values, because the format field has room to move within the column. Notice that the decimal points remain aligned but that the numbers shift in relation to the column width.

```
just=left               just=center              just=right

********                ********                 ********
123.45                   123.45                    123.45
234.50                   234.50                    234.50
   .                        .                         .
987.65                   987.65                    987.65
```

Now, if you add JUSTIFY=ON, then the values are formatted within the column without regard to the format width. The results are as follows:

```
justify=on              justify=on              justify=on
just=left               just=center             just=right

********                ********                ********
123.45                   123.45                   123.45
234.50                   234.50                   234.50
.                          .                        .
987.65                   987.65                   987.65
```

All destinations except LISTING justify the values in columns as if JUSTIFY=ON.

Formatting Values in Table Columns

The process of formatting the values in columns in a listing output is determined by the format of the variable and the values of three options: FORMAT=, FORMAT_WIDTH=, and FORMAT_NDEC=. It is a four-step process:

1. If you omit a FORMAT= option, then the format that the data component provides is used. If the data component does not provide a format, then ODS uses one of the following:
   - best8. for integers
   - D12.3 for doubles
   - the length of the variable for character variables
2. If a format width is specified in the FORMAT= option, then it will take precedence over the FORMAT_WIDTH= and FORMAT_NDEC= options.
3. If you specify a decimal width with the FORMAT= and FORMAT_NDEC= options, then the format that is specified with the FORMAT= option is used.
4. If you specify a format width with the FORMAT= and FORMAT_WIDTH= options, then the format that is specified with FORMAT= option is used.

The formatting attributes of a column is determined by the data component or the column template. This table summarizes the behavior of the column formatting attributes based on which attributes the column template provides.

Table 12.108 Summary of Column Formatting Attributes

Specifications Provided by the Column Template	Result
Nothing	Format name, width, and number of decimal places are determined by the data component.
Format name	Format name and width are determined by the column template; number of decimal places is determined by the data component.
Format name and width	Format name and width are determined by the column template.
Format name, width, and number of decimal places	All three are determined by the column template.

Specifications Provided by the Column Template	Result
Width	No name is specified; width is determined by the column template; number of decimal places is determined by the data component.
Number of decimal places	No name is specified; width is determined by the data component; number of decimal places is determined by the column template.

Examples: Modifying Tabular Output by Using the TEMPLATE Procedure

Example 1: Editing a Table Template That a SAS Procedure Uses

PROC TEMPLATE features:

EDIT statement

Header attributes:

JUST=
STYLE=

Table attributes:

DOUBLE_SPACE=
OVERLINE=
UNDERLINE=

Other ODS features:

ODS HTML statement

ODS SELECT statement

Data set: See "Creating the Exprev Data Set" on page 875.

Program Description

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

This example customizes the table template for the Moments output object from PROC UNIVARIATE. The first program uses the table template that SAS supplies to generate both listing output and HTML output of the Moments object.

The second program does the following:

- creates and edits a copy of the default table template
- edits a header within the table template
- sets column attributes to enhance the appearance of both the HTML and the listing output

Program 1: Using the Default Table Template That SAS Provides

Set the SAS system options. The OPTIONS statement controls several aspects of the listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file DefaultMoments-Body.htm in the current directory. Some browsers require an extension of .htm or .html on the filename.

```
ods html body="DefaultMoments-Body.htm ";
```

Select the output objects for the report. The ODS SELECT statement sends one output object, Moments, to the open ODS destinations. Both the LISTING and the HTML destinations are open. To learn the names of the output objects, run the procedure with the ODS TRACE ON statement in effect. For more information see
Example 1 on page 319.

```
ods select moments;
```

Compute the descriptive statistics for one variable. PROC UNIVARIATE computes the univariate statistics for one variable, Quantity. It uses the default table template, Base.Univariate.Moments from the template store Sashelp.Tmplmst.

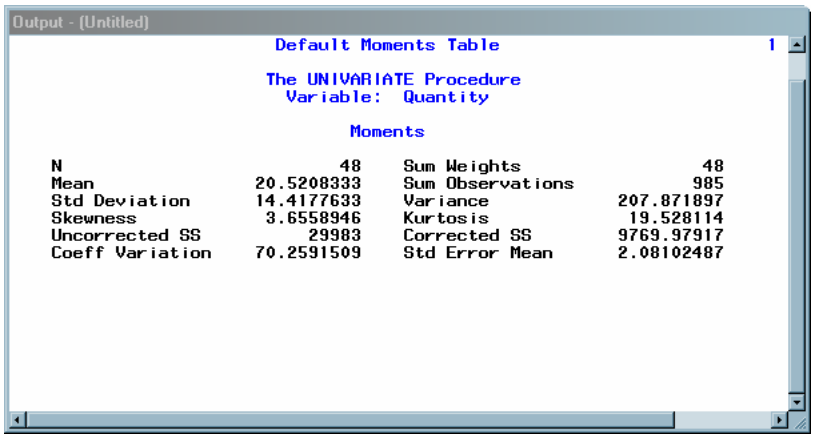
```
proc univariate data=exprev mu0=3.5;
   var Quantity;
title "Default Moments Table";
run;
```

Stop the creation of the HTML output. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser.

```
 ods html close;
```

Default Listing Output

Display 12.5 Listing Output from PROC UNIVARIATE (Default Moments Table)



Output - (Untitled)

Default Moments Table 1

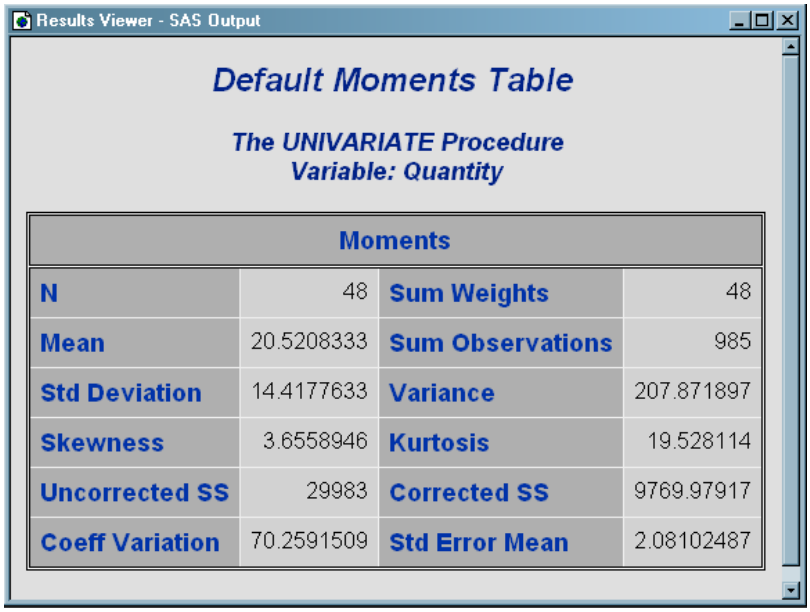
The UNIVARIATE Procedure
Variable: Quantity

Moments

N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

HTML Output from PROC UNIVARIATE (Default Moments Table)

Display 12.6 Default HTML Output



Results Viewer - SAS Output

Default Moments Table

The UNIVARIATE Procedure
Variable: Quantity

Moments			
N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Program 2: Using a Customized Table Template

Specify the search path in order to locate the table template. The ODS PATH statement specifies which locations to search for definitions or templates that were created by PROC TEMPLATE, as well as the order in which to search for them. The statement is included to ensure that the example works correctly. However, if you have not changed the path, you do not need to include this statement because it specifies the default path.

```
ods path sasuser.templat(update) sashelp.tmplmst(read);
```

Create a modified table template Base.Univariate.Moments. The EDIT statement looks in the available template stores for a table template called Base.Univariate.Moments. By default, it first looks in SASUSER.TEMPLAT, but it finds nothing. Next, it looks in Sashelp.Tmplmst, which contains the table templates that SAS provides. Because the EDIT statement can read this template, this is the one that it uses. The program does not specify a destination for the edited template, so PROC TEMPLATE writes to the first template store in the path that it can write to, which is SASUSER.TEMPLAT. Therefore, it creates a table template of the same name as the original one in SASUSER.TEMPLAT. See the
"ODS PATH Statement" on page 206.

(To learn the name of the table template that a procedure uses, run the procedure with the ODS TRACE ON statement in effect. See
"Example" on page 319.

```
proc template;
   edit base.univariate.moments;
```

Specify changes to the Moments output object. These three table attributes affect the presentation of the Moments output object in the listing output. They have no effect on its presentation in the HTML output. DOUBLE_SPACE= double spaces between the rows of the output object. OVERLINE= and UNDERLINE= draw a continuous line before the first row of the table and after the last row of the table.

```
   double_space=on;
   underline=on;
   overline=on;
```

Modify a table element. This EDIT statement edits the table element Head within the table template.

```
      edit head;
```

Modify the appearance of the header. The STYLE= attribute alters the style element that produces the Head table element. The style element Header is defined in the default style, Styles.Default. Many procedures, including PROC UNIVARIATE, use this style element to produce headers for tables and columns. (For information on viewing a style, see
"Styles That Are Shipped with SAS Software" on page 30.) In this case, the STYLE= attribute specifies green for the foreground color and italic for the font style. All other attributes that are included in Header remain in effect. The STYLE= attribute affects only the HTML output.

```
         style=header{color=green fontstyle=italic};
```

Left-justify the header text. The JUST= attribute left-justifies the text of the header in both the listing and the HTML output.

```
         just=left;
```

Stop the editing of the table element and the table template. The first END statement ends the editing of the table element Head. The second END statement ends the editing of the table Base.Univariate.Moments.

```
      end;
   end;
run;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file Custommoments-Body.htm in the current directory. Some browsers require an extension of .htm or .html on the filename.

```
ods html body="Custommoments-Body.htm";
```

Select the output objects for the report. The ODS SELECT statement sends one output object, Moments, to the open ODS destinations. Both the LISTING and the HTML destinations are open. To learn the names of the output objects, run the procedure with the ODS TRACE ON statement in effect. See
“Example” on page 319.

```
ods select moments;
```

Compute the descriptive statistics for one variable. PROC UNIVARIATE computes the univariate statistics for one variable, Quantity. This is the same PROC UNIVARIATE step that was used in “Program 1: Using the Default Table Template That SAS Provides” on page 757. The actual results of the procedure step are the same in this case, but they are presented differently because the procedure uses the edited table template. It does so because when it looks for Base.Univariate.Moments, it looks in the first template store in the path, SASUSER.TEMPLAT. If you wanted to use the table template that is supplied by SAS, you would have to change the path with the ODS PATH statement. For more information see the “ODS PATH Statement” on page 206.

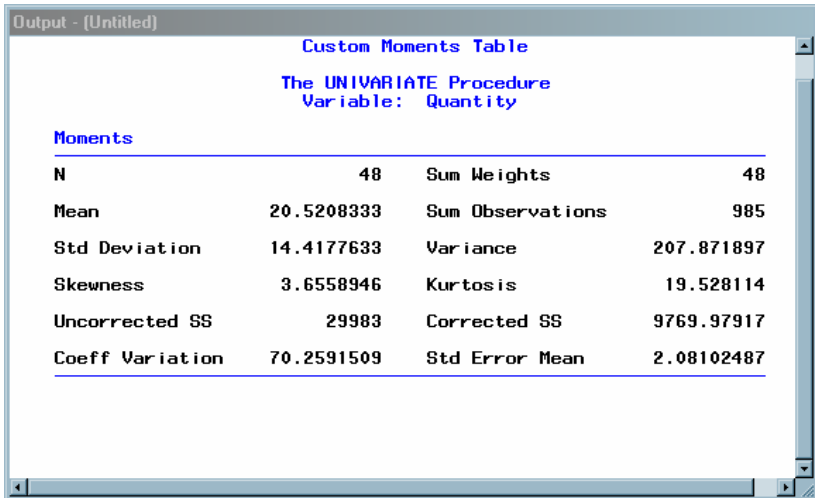
```
proc univariate data=exprev mu0=3.5;
   var Quantity;
title "Custom Moments Table";
run;
```

Stop the creation of the HTML output. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser.

```
ods html close;
```

Customized Listing Output

Display 12.7 Listing Output (Customized Moments Table) from PROC UNIVARIATE



Output - (Untitled)

Custom Moments Table

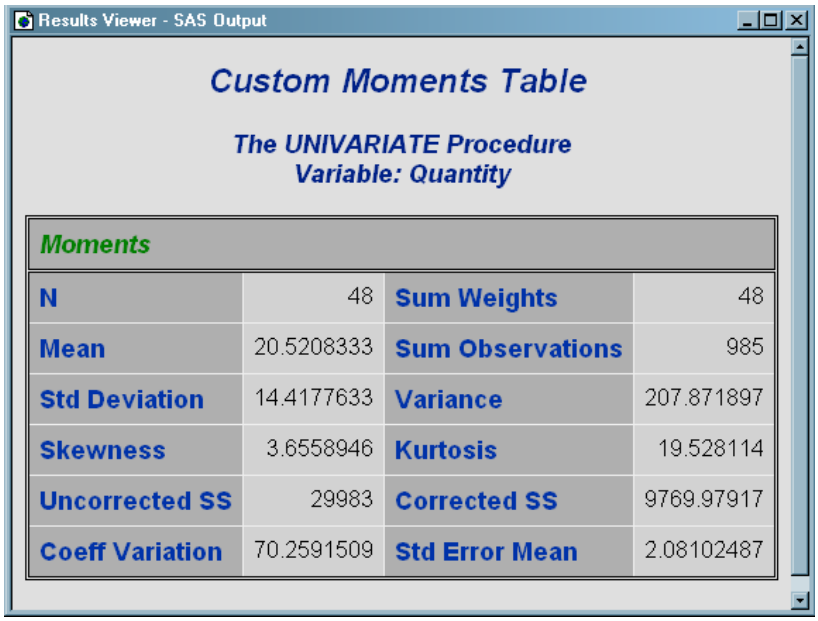
The UNIVARIATE Procedure
Variable: Quantity

Moments

N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Customized HTML Output

Display 12.8 Customized HTML Output (Customized Moments Table) from PROC UNIVARIATE (Viewed with Microsoft Internet Explorer)



Results Viewer - SAS Output

Custom Moments Table

The UNIVARIATE Procedure
Variable: Quantity

Moments			
N	48	Sum Weights	48
Mean	20.5208333	Sum Observations	985
Std Deviation	14.4177633	Variance	207.871897
Skewness	3.6558946	Kurtosis	19.528114
Uncorrected SS	29983	Corrected SS	9769.97917
Coeff Variation	70.2591509	Std Error Mean	2.08102487

Example 2: Comparing the EDIT Statement with the DEFINE TABLE Statement

PROC TEMPLATE features:

EDIT statement
COLUMN statement
DEFINE statement:
    STYLE= attribute
NOTES statement
DYNAMIC statement

Other ODS features:

ODS PATH statement
ODS HTML statement
ODS HTML CLOSE statement

Data set: See "Creating the Exprev Data Set" on page 875.

Program Description

This example compares the use of an EDIT statement with a DEFINE TABLE statement for the same table template. The first program uses the EDIT statement to change the Base.Summary table template. The foreground color of the NOBS column is changed to green. The other templates and attributes of the Base.Summary table template remain the same. The second program uses the DEFINE TABLE statement to define a new table using the same name, Base.Summary. The NOBS column is the only column defined in the new table template. When the PROC SUMMARY step executes, only the NOBS column is printed. The only style attribute that formats the column is the **`color=green`** attribute.

Program 1

Edit the existing table template Base.Summary. The ODS PATH statement specifies which item store to search first for the table template. The EDIT statement edits the table template Base.Summary. The modified table template Base.Summary is written to the WORK.TEMPLAT item store.

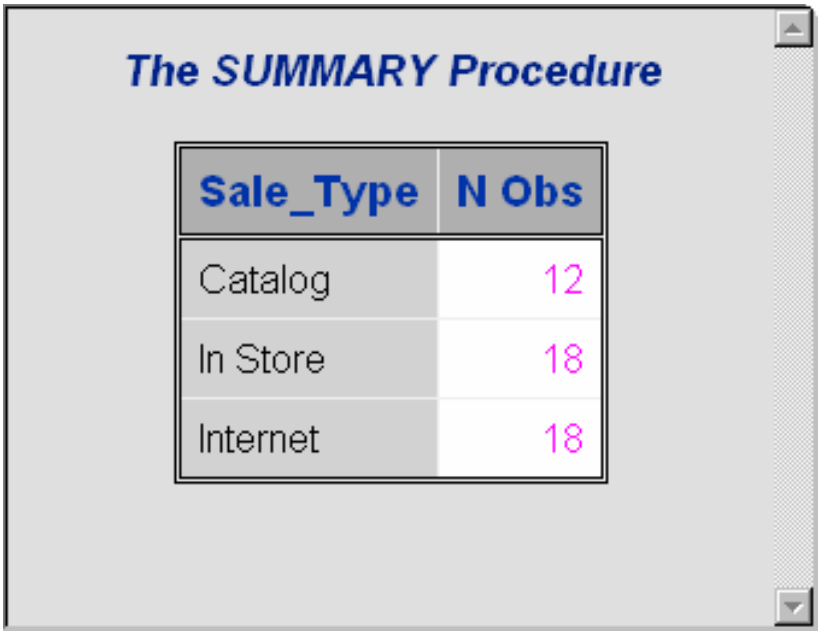
Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

```
ods path work.templat (update) sashelp.tmplmst (read);
proc template;
    edit Base.Summary;
    edit nobs;
    style={color=magenta background=white};
end;

end;
run;
```

```
ods html file="temp.html";

proc summary data=exprev print;
    class Sale_Type;
run;

ods html close;
```

Display 12.9 HTML Output Using an Edited Table Template for Base.Summary

The column labeled Age remains in the output because Age is defined as a dynamic variable which is passed to the original Base.Summary table template and Age is specified as the CLASS variable. The attributes of the NOBS column are modified in the EDIT statement where the NOBS column is defined.



Output 12.1 Base.Summary Table Template Modified by the EDIT Statement

The modified Base.Summary table template changes the foreground color of the NOBS column to green. The vertical alignment and heading of the NOBS column, and the other table attributes, are retained from the default table template and stay the same. To view the Base.Summary table template created by Program 1, follow these steps.

1 Submit this command in the command bar:

```
odstemplates
```

2 Double-click the item store **WORK.TEMPLAT**.

3 Double-click the item store **Base**.

4 Right-click the table template Summary and select **Open**. The table template Base.Summary is displayed in the Template Browser window.

```
proc template;
   define table Base.Summary / store = SASUSER.TEMPLAT;
      notes "Summary table for MEANS and SUMMARY";
      dynamic one_var one_var_label one_var_name clmpct;
      column class nobs id type ways (varname) (label) (min) (max) (range) (n)
         (nmiss) (sumwgt) (sum) (mean) (uss) (css) (var) (stddev) (cv) (stderr
         ) (t) (probt) (lclm) (uclm) (skew) (kurt) (median) (mode) (q1) (q3) (
         qrange) (p1) (p5) (p10) (p25) (p50) (p75) (p90) (p95) (p99);
      header h;
      define p99;
         header = "99th Pctl";
         generic;
      end;
      define p95;
         header = "95th Pctl";
         generic;
      end;
      define p90;
         header = "90th Pctl";
         generic;
      end;
      define p75;
         header = "75th Pctl";
         generic;
      end;
      define p50;
         header = "50th Pctl";
         generic;
      end;
      define p25;
         header = "25th Pctl";
         generic;
      end;
      define p10;
         header = "10th Pctl";
         generic;
      end;
      define p5;
         header = "5th Ptcl";
         generic;
      end;
      define p1;
         header = "1st Pctl";
         generic;
      end;
      define qrange;
         header = "Quartile Range";
         generic;
      end;
      define q3;
         header = "Upper Quartile";
         generic;
      end;
      define q1;
         header = "Lower Quartile";
         generic;
      end;
```

```
define mode;
        header = "Mode";
        generic;
     end;
      define median;
        header = "Median";
        generic;
     end;
      define kurt;
        header = "Kurtosis";
        generic;
     end;
     define skew;
        header = "Skewness";
        generic;
     end;
     define uclm;
        define header huclm;
           text "Upper " clmpct BEST8. %nrstr("%%/CL for Mean");
           split = "/";
        end;
        header = huclm;
        generic;
     end;
     define lclm;
        define header hlclm;
           text "Lower " clmpct BEST8. %nrstr("%%/CL for Mean");
           split = "/";
        end;
        header = hlclm;
        generic;
     end;
     define probt;
        parent = Common.ParameterEstimates.Probt;
        generic;
     end;
     define t;
        parent = Common.ParameterEstimates.tValue;
        generic;
     end;
     define stderr;
        header = "Std Error";
        parent = Common.ParameterEstimates.StdErr;
        generic;
     end;
     define cv;
        header = "Coeff of Variation";
        generic;
     end;
     define stddev;
        header = "Std Dev";
        generic;
     end;
     define var;
        header = "Variance";
        generic;
     end;
     define css;
        define header hcss;
           text2 "CSS";
           text "Corrected SS";
        end;
        header = hcss;
        generic;
     end;
     define uss;
        define header huss;
           text2 "USS";
           text "Uncorrected SS";
        end;
        header = huss;
        generic;
     end;
```

```
define mean;
      header = "Mean";
      generic;
   end;
   define sum;
      header = "Sum";
      generic;
   end;
   define sumwgt;
      header = "Sum Wgts";
      generic;
   end;
   define nmiss;
      header = "N Miss";
      generic;
   end;
   define n;
      header = "N";
      generic;
   end;
   define range;
      header = "Range";
      generic;
   end;
   define max;
      define header hmax;
         text2 "Max";
         text "Maximum";
      end;
      header = hmax;
      generic;
   end;
   define min;
      define header hmin;
         text2 "Min";
         text "Minimum";
      end;
      header = hmin;
      generic;
   end;
   define label;
      header = "Label";
      id;
      generic;
   end;
   define varname;
      header = "Variable";
      id;
      generic;
   end;
   define ways;
      header = "Ways";
      vjust = T;
      id;
   end;
   define type;
      header = "Type";
      vjust = T;
      id;
   end;
   define id;
      vjust = T;
      id;
      generic;
   end;
```

```
   define nobs;
         header = "N Obs";
         vjust = T;
         style = {
            color = green
            };
            id;
         end;
      define class;
         vjust = T;
         id;
         generic;
         blank_internal_dups;
      end;
      define h;
         text "Analysis Variable : " one_var_name " " one_var_label;
         space = 1;
         just = C;
         print = one_var;
         spill_margin;
      end;
      required_space = 5;
      underline;
      overline;
      byline;
      use_format_defaults;
      double_space;
      split_stack;
      use_name;
      order_data;
      classlevels;
   end;
run;
```

Program 2

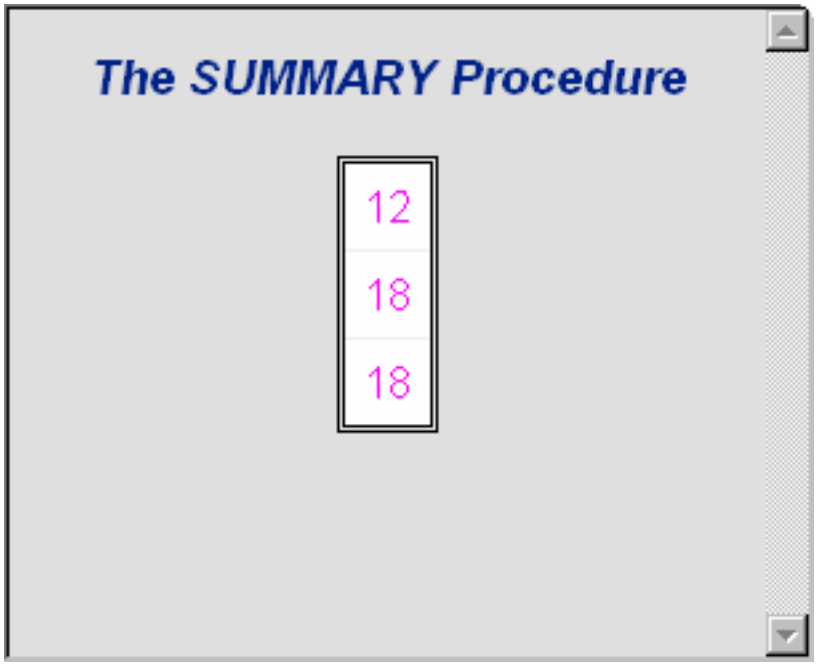
Define the table Base.Summary. The ODS PATH statement specifies which item store to search first for the table template. The DEFINE TABLE statement creates a new table template Base.Summary. The new table template Base.Summary is written to the WORK.TEMPLAT item store.

```
ods path work.templat (update) sashelp.tmplmst (read);
proc template;
   define table Base.Summary;
      notes "Summary table for MEANS and SUMMARY";
      dynamic clmpct one_var_name one_var_label one_var;
      column class nobs id type ways (varname) (label) (min) (max) (range) (n
         ) (nmiss) (sumwgt) (sum) (mean) (uss) (css) (var) (stddev) (cv) (
         stderr) (t) (probt) (lclm) (uclm) (skew) (kurt) (median) (mode) (q1)
         (q3) (qrange) (p1) (p5) (p10) (p25) (p50) (p75) (p90) (p95) (p99);


      define nobs;
        style={color=magenta backgroundcolor=white};

      end;

   end;
run;

ods html file="temp.html";
```

```
proc summary data=exprev print;
class Sale_Type;
run;

ods html close;
```

Display 12.10 HTML Output That Uses the Table Template Base.Summary.

The column labeled Age is missing because it was not defined in the new table template Base.Summary. The new table template only defined the NOBS column with a green foreground and no column headings.



Output 12.2 Base.Summary Table Template Created by the DEFINE TABLE Statement

The Base.Summary table template defines the foreground color of the NOBS column to green. Because the vertical alignment and heading of the NOBS column, and the other table attributes, are not defined, they are no longer part of the Base.Summary table template. To view the table template Base.Summary created by Program 2, follow these steps.

1 Submit this command:

```
odstemplates
```

2 Double-click the item store **WORK.TEMPLAT**.

3 Double-click the item store **Base**.

4 Right-click the table template **Summary** and select **Open**. The table template Base.Summary is displayed in the Template Browser window.

```
proc template;
   define table Base.Summary / store = WORK.TEMPLAT;
      notes "Summary table for MEANS and SUMMARY";
      dynamic clmpct one_var_name one_var_label one_var;
      column class nobs id type ways (varname) (label) (min)
      (max) (range) (n)(nmiss) (sumwgt) (sum) (mean) (uss) (css)
      (var) (stddev) (cv) (stderr) (t) (probt) (lclm) (uclm) (skew)
      (kurt) (median) (mode) (q1) (q3) (qrange) (p1) (p5) (p10)
      (p25) (p50) (p75) (p90) (p95) (p99);
      define nobs;
         style = {
            color = green
            };
      end;
   end;
run;
```

Example 3: Creating a New Table Template

PROC TEMPLATE features:

- Table attributes:
  - DOUBLE_SPACE=
  - OVERLINE=
  - UNDERLINE=
- DEFINE TABLE statement:
  - COLUMN statement
  - DEFINE statement (for columns):
    - GENERIC= attribute
    - HEADER= attribute
    - ID= attribute
    - STYLE= attribute
    - VJUST= attribute
  - DEFINE statement (for headers):
    - TEXT statement
    - STYLE= attribute
    - SPACE= attribute
  - DEFINE FOOTER statement

HEADER statement
MVAR statement

Other ODS features:
ODS HTML statement

FILE statement with ODS= option

PUT statement with _ODS_ argument

Data set: See “Creating the Charity Data Set” on page 869.

Program Description

This example creates a custom table template for an output data set that PROC MEANS produces.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 903. △

Program 1: Producing an Output Data Set with PROC MEANS

Set the SAS system options. The OPTIONS statement controls several aspects of the listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Create formats for the variables Year and School. PROC FORMAT creates formats for Year and School.

```
proc format;
   value yrFmt . = " All";
   value $schFmt " " = "All    ";
   run;
```

Compute the descriptive statistics, and specify the options and subgroups for analysis. This PROC MEANS step analyzes the data for the one-way combination of the class variables and across all observations. It creates an output data set that includes variables for the total and average amount of money raised. The data set also includes new variables for the top three amounts of money raised, the names of the three students who raised the money, the years when the students raised the money, and the schools that the students attended.

```
proc means data=Charity descendTypes charType noprint;
   class School Year;
   var moneyRaised;
   types () School year;
   output out=top3list sum= mean=
      idgroup ( max(moneyRaised) out[3](moneyRaised name school year)= )
      / autoname;
   run;
```

Print the report. This PROC PRINT step generates traditional listing output of the output data set that PROC MEANS created.

```
proc print data=top3list noobs;
   title "Simple PROC PRINT of the Output Data Set";
run;
```

Listing Output from PROC PRINT

Output 12.3 PROC PRINT Listing Output from PROC MEANS

```
               Simple PROC PRINT of the Output Data Set              1

                              money    money
                            Raised_  Raised_  money    money    money
School  Year _TYPE_ _FREQ_      Sum     Mean Raised_1 Raised_2 Raised_3

Kennedy  All   10      53  $1575.95   $29.73   $72.22   $52.63   $43.89
Monroe   All   10      56  $1616.80   $28.87   $78.65   $65.44   $56.87
All     1992   01      31   $892.92   $28.80   $55.16   $53.76   $52.63
All     1993   01      32   $907.92   $28.37   $65.44   $47.33   $42.23
All     1994   01      46  $1391.91   $30.26   $78.65   $72.22   $56.87
All      All   00     109  $3192.75   $29.29   $78.65   $72.22   $65.44



Name_1  Name_2  Name_3  School_1 School_2 School_3 Year_1 Year_2 Year_3

Luther  Thelma  Jenny   Kennedy  Kennedy  Kennedy   1994   1992   1992
Willard Cameron L.T.    Monroe   Monroe   Monroe    1994   1993   1994
Tonya   Edward  Thelma  Monroe   Monroe   Kennedy   1992   1992   1992
Cameron Myrtle  Bill    Monroe   Monroe   Kennedy   1993   1993   1993
Willard Luther  L.T.    Monroe   Kennedy  Monroe    1994   1994   1994
Willard Luther  Cameron Monroe   Kennedy  Monroe    1994   1994   1993
```

Program 2: Building a Custom Table Template for the TopN Report

Set the SAS system options. The OPTIONS statement controls several aspects of the listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file Topn-Body.htm in the current directory. Some browsers require an extension of .htm or .html on the filename.

```
ods html body="Topn-Body.htm";
```

Create the table template Means.Topn The DEFINE statement creates the table template Means.Topn in the first template store in the path for which you have write access. By default, this template store is SASUSER.TEMPLAT.

```
proc template;
   define table means.topn;
```

Specify the symbols that reference three macro variables. The MVAR statement defines three symbols that reference macro variables. ODS will use the values of these variables as strings. References to the macro variables are resolved when ODS binds the template and the data component to produce an output object. First_Year and Last_Year will contain the values of the first and last years for which there are data. Their values are assigned by the SYMPUT statements in the DATA step. SYSDATE9 is an automatic macro variable whose value is always available.

```
mvar first_year last_year sysdate9;
```

Specify the column names and the order in which they appear in the report. The COLUMN statement declares these variables as columns in the table and specifies their order in the table. If a column name appears in parentheses, then PROC TEMPLATE stacks the values of all variables that use that column template one below the other in the output object. Variables are assigned a column template in the DATA step that appears later in the program.

```
column class sum mean (raised) (name) (school) (year);
```

Specify three customized changes to the table template. These three table attributes affect the presentation of the output object in the listing output. They have no effect on its presentation in the HTML output. DOUBLE_SPACE= double spaces the rows of the output object. OVERLINE= and UNDERLINE= draw a continuous line before the first row of the table and after the last row of the table.

```
 double_space=on;
 overline=on;
 underline=on;
```

Specify the two table headers and the order in which they appear in the report. The HEADER statement declares Table_Header_1 and Table_Header_2 as headers in the table and specifies the order in which the headers appear in the output object.

```
header table_header_1 table_header_2;
```

Create the table element Table_Header_1. The DEFINE statement and its substatement and attribute define Table_Header_1. The TEXT statement specifies the text of the header. The STYLE= attribute alters the style element that displays the table header. The style element Header is defined in the default style, Styles.Default. (For information on viewing a style, see “Styles That Are Shipped with SAS Software” on page 30.) In this case, the STYLE= attribute specifies a large font size. All other attributes that are included in Header remain in effect. This attribute affects only the HTML output.

The END statement ends the header template.

```
define table_header_1;
   text "Top Three Fund Raisers";
   style=header{fontsize=6};
end;
```

Create the table element Table_Header_2. The DEFINE statement and its substatement and attribute define Table_Header_2. The TEXT statement uses text and the macro variables First_Year and Last_Year to specify the contents of the header. When ODS binds the data component to the table template (in the DATA step that follows), it will resolve the values of the macro variables First_Year and Last_Year. The table template itself contains references to the macro variables.

The SPACE= attribute inserts a blank line after the header (in the listing output only).

The END statement ends the header template.

```
define table_header_2;
    text "from " first_year " to " last_year;
    space=1;
end;
```

Create the table element Table_Footer. The DEFINE statement and its substatement and attribute define Table_Footer. The FOOTER argument declares Table_Footer as a footer. (Compare this approach with the creation of the headers. You could use a FOOTER statement instead of the FOOTER argument in the DEFINE statement.)

The TEXT statement specifies the text of the footer. When ODS binds the data component to the table template (in the DATA step that follows), it will resolve the value of the macro variable SYSDATE9. The table template itself contains a reference to the macro variable. The SPLIT= attribute specifies the asterisk as the split character. This prevents the header from splitting at the open parenthesis. If no split character is specified, then ODS interprets the nonalphabetic, leading character as the split character (see the discussion of *text-specification(s)* in "TEXT Statement" on page 639.) Alternatively, place a space character before the open parenthesis.

The STYLE= attribute alters the style element that displays the table footer. The style element Header is defined in the default style, Syles.Default. (For information on viewing a style, see "Viewing the Contents of a Style" on page 538.) In this case, the STYLE= attribute specifies a small font size. All other attributes that are included in Footer remain in effect. This attribute affects only the HTML output.

The END statement ends the footer template.

```
define footer table_footer;
   text "(report generated on " sysdate9 ")";
   split="*";
   style=header{fontsize=2};
end;
```

Create the column template Class. The DEFINE statement and its attributes create the column template Class. (The COLUMN statement earlier in the program declared Class as a column.)

GENERIC= specifies that multiple variables can use the same column template. GENERIC= is not specific to a destination.

ID= specifies that this column should be repeated on every data panel if the report uses multiple data panels. ID= affects only the listing output.

VJUST= specifies that the text appear at the top of the HTML table cell that it is in. VJUST= affects only the HTML output.

STYLE= specifies that the column uses the DATA table element. This table element is defined in the default style, which is the style that is being used. STYLE= affects only the HTML output.

The END statement ends the template.

Notice that, unlike subsequent column templates, this column template does not include a header. This is because the same header is not appropriate for all the variables that use this column template. Because there is no header specified here or in the FILE statement, the header comes from the label that was assigned to the variable in the DATA step.

```
define class;
   generic=on;
   id=on;
   vjust=top;
   style=data;
end;
```

Create six additional columns. Each of these DEFINE statements and its attributes creates a column template. GENERIC= specifies that multiple variables can use a column template (although in the case of Sum and Mean, only one variable uses the template). HEADER= specifies the text for the column header. VJUST= specifies that the text appear at the top of the HTML table cell that it is in. The END statement ends the template.

```
define sum;
   generic=on;
   header="Total Dollars Raised";
   vjust=top;
end;

define mean;
   generic=on;
   header="Average Dollars per Student";
   vjust=top;
end;

define raised;
   generic=on;
   header="Individual Dollars";
end;

define name;
   generic=on;
   header="Student";
end;

define school;
   generic=on;
```

```
      header="School";
   end;

   define year;
      generic=on;
      header="Year";
   end;
```

End the table template. This END statement ends the table template. The RUN statement ends the PROC TEMPLATE step.

```
   end;
run;
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object. The SET statement reads the data set TOP3LIST that was created with PROC MEANS.

```
data _null_;
   set top3list;
```

Route the DATA step results to ODS and use the Means.Topn table template. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information on using the DATA step with ODS, see Chapter 3, "Output Delivery System and the DATA Step," on page 39.) The TEMPLATE= suboption tells ODS to use the table template named Means.Topn, which was previously created with PROC TEMPLATE.

```
   file print ods = (
      template="means.topn"
```

Specify the column template to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table template. For example, the first *column-specification* specifies that the first column of the output object contains the values of the variable SCHOOL and that it uses the column template named Class. GENERIC= must be set to ON in both the table template and each column assignment in order for multiple variables to use the same column template.

```
      columns=(
         class=school(generic=on)
         class=year(generic=on)
         sum=moneyRaised_sum(generic=on)
         mean=moneyRaised_mean(generic=on)
         raised=moneyRaised_1(generic=on)
         raised=moneyRaised_2(generic=on)
         raised=moneyRaised_3(generic=on)
         name=name_1(generic=on)
         name=name_2(generic=on)
         name=name_3(generic=on)
         school=school_1(generic=on)
         school=school_2(generic=on)
         school=school_3(generic=on)
         year=year_1(generic=on)
         year=year_2(generic=on)
         year=year_3(generic=on)
         )
```

```
);
```

Write the data values to the data component. The _ODS_ option and the PUT statement write the data values for all columns to the data component.

```
put _ods_;
run;
```

Stop the creation of HTML output. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser.

```
ods html close;
```

listing output for the TopN Report

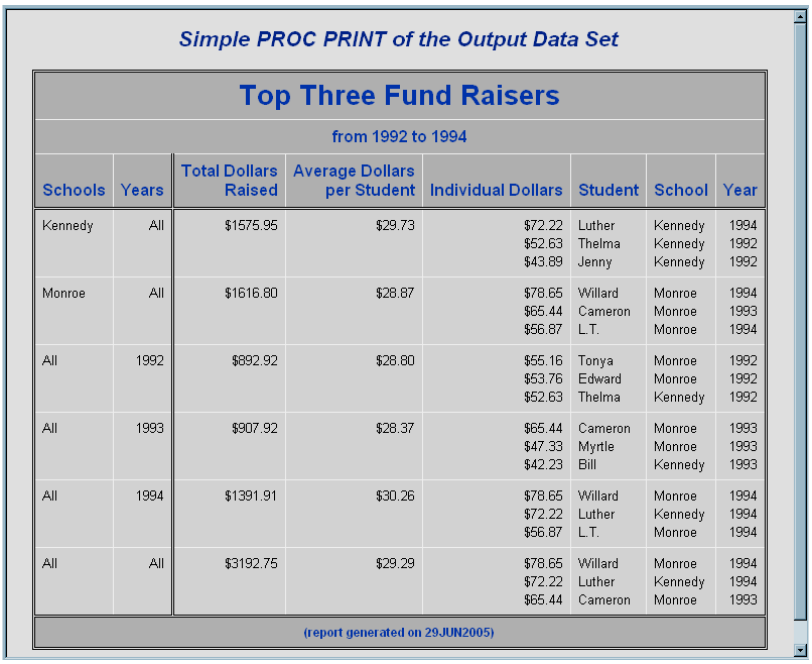
Compare this customized output to the PROC PRINT listing output in Output 12.3.

Output 12.4 Using a Customized Table to Produce Listing Output

```
  Simple PROC PRINT of the Output Data Set              1
                       Top Three Fund Raisers
                             from  to

                          Average
                    Total Dollars
                  Dollars     per  Individual
Schools  Years     Raised  Student     Dollars  Student  School   Year
_____________________________________________________________________

Kennedy    All  $1575.95    $29.73      $72.22  Luther   Kennedy  1994
                                       $52.63  Thelma   Kennedy  1992
                                       $43.89  Jenny    Kennedy  1992

Monroe     All  $1606.80    $28.69      $78.65  Willard  Monroe   1994
                                       $65.44  Cameron  Monroe   1993
                                       $56.87  L.T.     Monroe   1994

All       1992   $882.92    $28.48      $55.16  Tonya    Monroe   1992
                                       $53.76  Edward   Monroe   1992
                                       $52.63  Thelma   Kennedy  1992

All       1993   $907.92    $28.37      $65.44  Cameron  Monroe   1993
                                       $47.33  Myrtle   Monroe   1993
                                       $42.23  Bill     Kennedy  1993

All       1994  $1391.91    $30.26      $78.65  Willard  Monroe   1994
                                       $72.22  Luther   Kennedy  1994
                                       $56.87  L.T.     Monroe   1994

All        All  $3182.75    $29.20      $78.65  Willard  Monroe   1994
                                       $72.22  Luther   Kennedy  1994
                                       $65.44  Cameron  Monroe   1993
_____________________________________________________________________
                   (report generated on 30JUN2003)
```

HTML Output: Using a Customized Table for the TopN Report

Display 12.11 HTML Output for the TopN Report (Viewed with Microsoft Internet Explorer)



Simple PROC PRINT of the Output Data Set

Top Three Fund Raisers

from 1992 to 1994

Schools	Years	Total Dollars Raised	Average Dollars per Student	Individual Dollars	Student	School	Year
Kennedy	All	$1575.95	$29.73	$72.22	Luther	Kennedy	1994
				$52.63	Thelma	Kennedy	1992
				$43.89	Jenny	Kennedy	1992
Monroe	All	$1616.80	$28.87	$78.65	Willard	Monroe	1994
				$65.44	Cameron	Monroe	1993
				$56.87	L.T.	Monroe	1994
All	1992	$892.92	$28.80	$55.16	Tonya	Monroe	1992
				$53.76	Edward	Monroe	1992
				$52.63	Thelma	Kennedy	1992
All	1993	$907.92	$28.37	$65.44	Cameron	Monroe	1993
				$47.33	Myrtle	Monroe	1993
				$42.23	Bill	Kennedy	1993
All	1994	$1391.91	$30.26	$78.65	Willard	Monroe	1994
				$72.22	Luther	Kennedy	1994
				$56.87	L.T.	Monroe	1994
All	All	$3192.75	$29.29	$78.65	Willard	Monroe	1994
				$72.22	Luther	Kennedy	1994
				$65.44	Cameron	Monroe	1993

(report generated on 29JUN2005)

Example 4: Setting the Style Element for Cells Based on Their Values

PROC TEMPLATE features:

DEFINE TABLE statement:

- NMVAR statement
- NOTES statement
- TRANSLATE INTO statement

DEFINE COLUMN statement:

- BLANK_DUPS= attribute
- CELLSTYLE AS statement
- GENERIC= attribute

Other ODS features:

ODS HTML statement

FILE statement with ODS= option

PUT statement with _ODS_ argument

Data set: See “Creating the Grain_Production Data Set” on page 878.

Format: See “Creating the $CNTRY Format” on page 869.

Program Description

This example creates a template that uses different colors and font attributes for the text inside cells, depending on their values.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, "ODS HTML Statements for Running Examples in Different Operating Environments," on page 903. △

Program

Set the SAS system options. The OPTIONS statement controls several aspects of the listing output. None of these options affects the HTML output. The TITLE statement specifies a title.

```
options nodate pageno=1 pagesize=60 linesize=72;
title "Leading Grain Producers";
```

Create the table template Shared.Cellstyle. The DEFINE statement creates the table template Shared.Cellstyle in the first template store in the path that is available to write to. By default, this template store is SASUSER.TEMPLAT.

```
proc template;
   define table shared.cellstyle;
```

Specify that missing values show the text "No data" in the report. The TRANSLATE INTO statement translates missing values (.) into the string **No data**.

```
    translate _val_=. into "No data";
```

Store the information about the table in the table template. The NOTES statement provides information about the table. NOTES statements remain a part of the compiled table template whereas SAS comments do not.

```
    notes "NMVAR defines symbols that will be used to determine the colors
  of the cells.";
```

Specify the symbols that reference three macro variables. The NMVAR statement defines three symbols that reference macro variables. ODS will convert the variable's values to numbers (stored as doubles) before using them. References to the macro variables are resolved when ODS binds the template and the data component to produce an output object. The text inside quotation marks provides information about the symbols. This information becomes a part of the compiled table template whereas SAS comments do not.

LOW, MEDIUM, and HIGH will contain the values to use as the determinants of the style element that displays the cell. The values are provided just before the DATA step that produces the report.

```
   nmvar low "Use default style."
         medium "Use yellow foreground color and bold font weight"
         high "Use red foreground color and a bold, italic font.";
```

Control the repetition of values that do not change from one row to the next row. The CLASSLEVELS= attribute suppresses the display of the value in a column that is marked with BLANK_DUPS=ON if the value changes in a previous column that is also marked with BLANK_DUPS=ON. Because BLANK_DUPS= is set in a generic column, set this attribute as well.

```
    classlevels=on;
```

Create the column template Char_Var. The DEFINE statement and its attributes create the column template Char_Var. GENERIC= specifies that multiple variables can use the same column template. BLANK_DUPS= suppresses the display of the value in the column if it does not change from one row to the next (and, because CLASSLEVELS=ON for the table, if no value changes in a preceding column that is marked with BLANK_DUPS=ON changes).

The END statement ends the template.

```
    define column char_var;
       generic=on;
       blank_dups=on;
    end;
```

Create the column template Num_Var. The DEFINE statement and its attributes create the column template Num_Var. GENERIC= specifies that multiple variables can use the same column template.

```
    define column num_var;
       generic=on;
```

Align the values in the column without regard to the format field. JUSTIFY= justifies the values in the column without regard to the format field. For numeric variables, the default justification is RIGHT, so even the translated character value **No data** that is used for missing values is right-justified. Without JUSTIFY=ON in this column template, the value **No data** is formatted as a character variable (left-justified) within a format field that has the same width as the column.

```
       justify=on;
```

Specify which style element and style attributes to use for different values in the column. The CELLSTYLE AS statement specifies the style element and style attributes to use for different values in this column. If a value is less than or equal to the value of the variable LOW, the cell uses the unaltered Data style element. If a value is greater than LOW but less than or equal to the value of MEDIUM, the cell uses the style element Data with a foreground color of green and an italic font. Similarly, other values use a foreground color of yellow or red and combinations of a bold font weight and an italic font style. The CELLSTYLE AS statement affects only the HTML destination.

The END statement ends the column template.

```
       cellstyle _val_ <= low as data,
                 _val_ <= medium as data
                          {color=green fontstyle=italic},
                 _val_ <= high as data
                          {color=yellow fontweight=bold},
                      1 as data
                          {color=red fontstyle=italic
                           fontweight=bold};
    end;
```

End the table template. This END statement ends the table template. The RUN statement ends the PROC TEMPLATE step.

```
    end;
run;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file CellStyle-Body.htm in the current directory. Some browsers require an extension of .htm or .html on the filename.

```
ods html body="CellStyle-Body.htm";
```

Assign values to three macro variables. The %LET statements assign values to the macro variables LOW, MEDIUM, and HIGH.

```
%let low=10000;
%let medium=50000;
%let high=100000;
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component, and, eventually, an output object. The SET statement reads the data set Grain_Production.

```
data _null_;
   set grain_production;
```

Route the DATA step results to ODS and use the Shared.CellStyle table template. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information on using the DATA step with ODS, see Chapter 3, "Output Delivery System and the DATA Step," on page 39.) The TEMPLATE= suboption tells ODS to use the table template named Shared.CellStyle, which was previously created with PROC TEMPLATE.

```
    file print ods=(
         template="shared.cellstyle"
```

Specify the column template to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table template. For example, the first *column-specification* specifies that the first column of the output object contains the values of the variable YEAR and that it uses the column template named Char_Var. GENERIC= must be set to ON, both in the table template and in each column assignment, in order for multiple variables to use the same column template.

```
         columns=(
            char_var=year(generic=on)
            char_var=country(generic=on format=$cntry.)
            char_var=type(generic=on)
            num_var=kilotons(generic=on format=comma12.)
            )
         );
```

Write the data values to the data component. The _ODS_ option and the PUT statement write the data values for all columns to the data component.

```
   put _ods_;
run;
```

Stop the creation of HTML output. The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. Close the destination so that you can view the output with a browser.

```
ods html close;
```

Listing Output of a Customized Table

Output 12.5 Listing Output

Only the table customizations appear in the listing output. Table customizations include the suppression of values that do not change from one row to the next and the translation of missing values to **No data**. The style customizations that are specified in the CELLSTYLE AS statement do not appear in the listing output.

Leading Grain Producers 1

Year	Country	Type	Kilotons
1995	Brazil	Corn	36,276
		Rice	11,236
		Wheat	1,516
	China	Corn	112,331
		Rice	185,226
		Wheat	102,207
	India	Corn	9,800
		Rice	122,372
		Wheat	63,007
	Indonesia	Corn	8,223
		Rice	49,860
		Wheat	No data
	United States	Corn	187,300
		Rice	7,888
		Wheat	59,494
1996	Brazil	Corn	31,975
		Rice	10,035
		Wheat	3,302
	China	Corn	119,350
		Rice	190,100
		Wheat	109,000
	India	Corn	8,660
		Rice	120,012
		Wheat	62,620
	Indonesia	Corn	8,925
		Rice	51,165
		Wheat	No data
	United States	Corn	236,064
		Rice	7,771
		Wheat	62,099

HTML Output of a Customized Table

Display 12.12 HTML Output (Viewed with Microsoft Internet Explorer)

Both the table customizations and the style customizations appear in the HTML output. Table customizations include the suppression of values that do not change from one row to the next, and the translation of missing values to **`No data`**. The style customizations include the colors and font styles that are specified in the CELLSTYLE AS statement.

Leading Grain Producers

Year	Country	Type	Kilotons
1995	Brazil	Corn	36,276
		Rice	11,236
		Wheat	1,516
	China	Corn	112,331
		Rice	185,226
		Wheat	102,207
	India	Corn	9,800
		Rice	122,372
		Wheat	63,007
	Indonesia	Corn	8,223
		Rice	49,860
		Wheat	No data
	United States	Corn	187,300
		Rice	7,888
		Wheat	59,494
1996	Brazil	Corn	31,975

Example 5: Setting the Style Element for a Specific Column, Row, and Cell

PROC TEMPLATE features:

DEFINE STYLE statement:

REPLACE statement

DEFINE TABLE statement:

CELLSYTLE–AS statement
DEFINE COLUMN statement:
DEFINE HEADER statement:
TEXT statement
DEFINE HEADER statement:
TEXT statement

Other ODS features:
FILE statement with ODS= option
ODS HTML statement:
STYLE= option
ODS PDF statement:
STYLE= option
PUT statement with _ODS_ argument
ODS TRACE statement

Data set: See "Creating the Exprev Data Set" on page 875.

Program Description

This example combines a customized style with a customized table template to produce output with a checkerboard pattern of table cells.

Program

```
options obs=20;
title;
```

Create the new style Greenbar. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style Greenbar.

```
proc template;
   define style greenbar;
```

Specify the parent style from which the Greenbar style inherits its attributes. The PARENT= attribute specifies the style from which the Greenbar definition inherits its style elements and attributes. All the style elements and their attributes that are specified in the parent's definition are used in the current definition unless the current definition overrides them.

```
    parent=styles.printer;
```

Change the colors used in the headers and footers. The REPLACE statement adds a style element to the Greenbar style from the parent style, but the background is light green, the foreground is black, and the font is bold and has a size of 3.

```
    replace headersandfooters from cell /
          backgroundcolor=light green
```

```
        color=black
        fontsize=3
        fontweight=bold
      ;
```

End the style. The END statement ends the style. The RUN statement executes the PROC TEMPLATE step.

```
    end;
 run;
```

Create the HTML and PDF output and specify the style that you want to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the file greenbar.html in the current directory. The STYLE= option tells ODS to use Greenbar as the style when it formats the output.

The ODS PDF statement opens the PDF destination and creates PDF output. It sends all output objects to the file greenbar.pdf in the current directory. The STYLE= option tells ODS to use Greenbar as the style when it formats the output.

```
ods html body="greenbar.html" style=greenbar;
ods pdf file="greenbar.pdf" style=greenbar;
```

Create the table template Checkerboard. The DEFINE statement creates the table template Checkerboard in the first template store in the path that is available to write to. By default, this template store is SASUSER.TEMPLAT.

```
proc template;
   define table Checkerboard;
```

Specify which style element and style attributes to use for different cells.

The CELLSTYLE-AS statement specifies the style element and style attributes to use for cells in each of the rows and columns. The CELLSTYLE-AS statement creates the checkerboard effect in the output. If both the row and column are odd numbered, then the cell is yellow in color. Similarly, if both the row and column are even numbered, then the cell is yellow in color. The CELLSTYLE-AS statement has no effect on the LISTING destination because it is changing style elements and style attributes which have no effect in listing output.

```
cellstyle mod(_row_,2) && mod(_col_,2) as data{backgroundcolor=yellow fontweight=bold },
               not(mod(_row_,2)) && not(mod(_col_,2)) as data{backgroundcolor=yellow fontwei
               1 as data;
```

Create the header template Top. The DEFINE HEADER statement defines the table header Top.

The TEXT statement specifies the text of the header “Checkerboard Table Template”.

The END statement ends the header template.

```
define header top;
   text "Checkerboard Table Template";
```

```
end;
```

Create the column template Country. The DEFINE COLUMN statement creates the column template Country.
The DEFINE HEADER statement creates the header template Bar.
The DATANAME= column attribute specifies the name of the column Country in the data component to associate with the column template Country.
The TEXT statement specifies the text to use in the header.
The first END statement ends the header template.
The HEADER statement declares Bar as the header in the table.
The second END statement ends the column template.

```
define column country;
   dataname=country;
   define header bar;
      text "Country";
   end;
   header=bar;
end;
```

Create the column template OrderDate. The DEFINE COLUMN statement creates the column template OrderDate.
The DATANAME= column attribute specifies the name of the column OrderDate in the data component to associate with the column template OrderDate.
The DEFINE HEADER statement creates the header template Bar.
The TEXT statement specifies the text “Order Date” to use in the header.
The first END statement ends the header template.
The HEADER statement declares Bar as the header in the table.
The second END statement ends the column template.

```
define column OrderDate;
   dataname=Order_Date;
   define header bar;
      text "Order Date";
   end;
   header=bar;
end;
```

Create the column template ShipDate. The DEFINE COLUMN statement creates the column template ShipDate.
The DATANAME= column attribute specifies the name of the column template ShipDate in the data component to associate with the column template ShipDate.
The DEFINE HEADER statement creates the header template Bar.
The TEXT statement specifies the text “Ship Date” to use in the header.
The first END statement ends the header template.
The HEADER statement declares Bar as the header in the table.
The second END statement ends the column template.

```
define column ShipDate;
   dataname=Ship_Date;
```

```
      define header bar;
         text "Ship Date";
      end;
      header=bar;
   end;
```

Create the column template SaleType. The DEFINE COLUMN statement creates the column template SaleType.

The DATANAME= column attribute specifies the name of the column template SaleType in the data component to associate with the column template SaleType.

The DEFINE HEADER statement creates the header template Bar.

The TEXT statement specifies the text "Sale Type" to use in the header.

The first END statement ends the header template.

The HEADER statement declares Bar as the header in the table.

The second END statement ends the column template.

```
   define column SaleType;
      dataname=Sale_Type;
      define header bar;
         text "Sale Type";
      end;
      header=bar;
   end;
```

End the table template. The END statement ends the table template. The RUN statement executes the TEMPLATE procedure.

```
end;
run;
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component that is used to produce an output object.

The SET statement reads the data set Work.Exprev.

```
data _null_;
   set work.exprev;
```

Route the DATA step results to ODS and use the Checkerboard table template. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, "Output Delivery System and the DATA Step," on page 39.) The TEMPLATE= suboption tells ODS to use the table template named Checkerboard.

```
file print ods=(template="Checkerboard");
  put _ods_;
run;
```

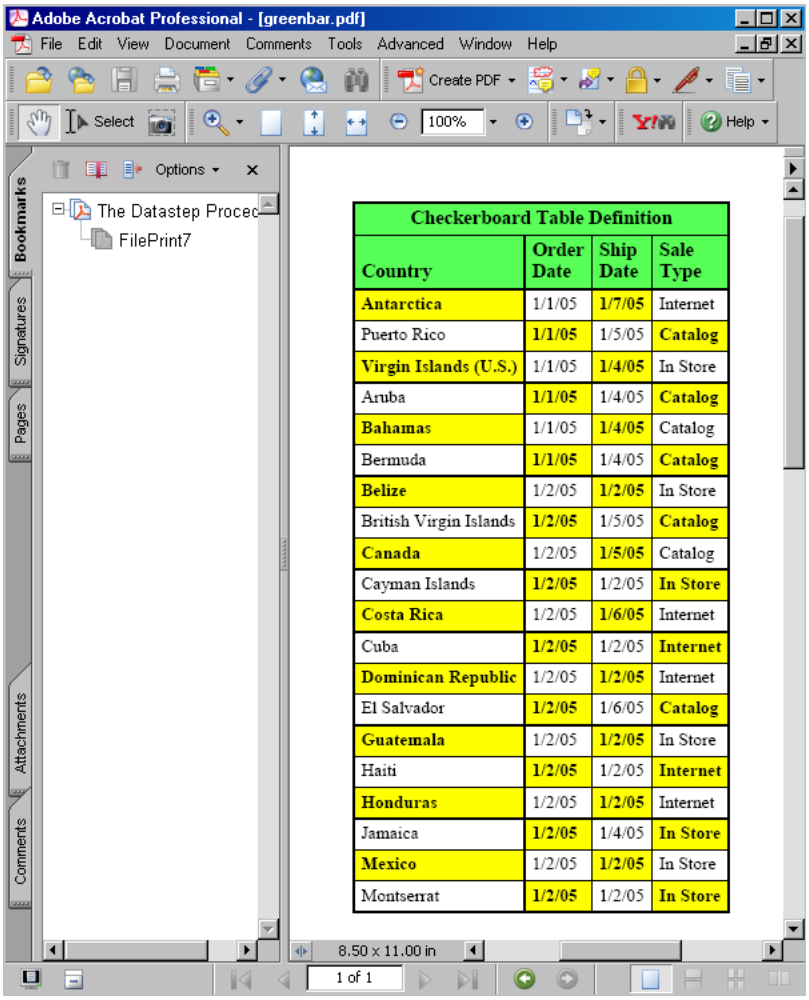
Stop the creation of HTML and PDF output. The ODS HTMLCLOSE statement closes the HTML destination and all the files that are associated with it. The ODS PDf CLOSE statement closes the PDF destination and all the files that are associated with it. You must close the destinations before you can view the output.

```
ods html close;
ods pdf close;
```

Display 12.13 HTML Output (Viewed with Internet Explorer 6.0)

Checkerboard Table Definition			
Country	**Order Date**	**Ship Date**	**Sale Type**
Antarctica	1/1/05	**1/7/05**	Internet
Puerto Rico	**1/1/05**	1/5/05	**Catalog**
Virgin Islands (U.S.)	1/1/05	**1/4/05**	In Store
Aruba	**1/1/05**	1/4/05	**Catalog**
Bahamas	1/1/05	**1/4/05**	Catalog
Bermuda	**1/1/05**	1/4/05	**Catalog**
Belize	1/2/05	**1/2/05**	In Store
British Virgin Islands	**1/2/05**	1/5/05	**Catalog**
Canada	1/2/05	**1/5/05**	Catalog
Cayman Islands	**1/2/05**	1/2/05	**In Store**
Costa Rica	1/2/05	**1/6/05**	Internet
Cuba	**1/2/05**	1/2/05	**Internet**
Dominican Republic	1/2/05	**1/2/05**	Internet
El Salvador	**1/2/05**	1/6/05	**Catalog**
Guatemala	1/2/05	**1/2/05**	In Store
Haiti	**1/2/05**	1/2/05	**Internet**
Honduras	1/2/05	**1/2/05**	Internet
Jamaica	**1/2/05**	1/4/05	**In Store**
Mexico	1/2/05	**1/2/05**	In Store
Montserrat	**1/2/05**	1/2/05	**In Store**

Display 12.14 PDF Output (Viewed with Acrobat Reader 5.0)



Checkerboard Table Definition

Country	Order Date	Ship Date	Sale Type
Antarctica	1/1/05	1/7/05	Internet
Puerto Rico	1/1/05	1/5/05	Catalog
Virgin Islands (U.S.)	1/1/05	1/4/05	In Store
Aruba	1/1/05	1/4/05	Catalog
Bahamas	1/1/05	1/4/05	Catalog
Bermuda	1/1/05	1/4/05	Catalog
Belize	1/2/05	1/2/05	In Store
British Virgin Islands	1/2/05	1/5/05	Catalog
Canada	1/2/05	1/5/05	Catalog
Cayman Islands	1/2/05	1/2/05	In Store
Costa Rica	1/2/05	1/6/05	Internet
Cuba	1/2/05	1/2/05	Internet
Dominican Republic	1/2/05	1/2/05	Internet
El Salvador	1/2/05	1/6/05	Catalog
Guatemala	1/2/05	1/2/05	In Store
Haiti	1/2/05	1/2/05	Internet
Honduras	1/2/05	1/2/05	Internet
Jamaica	1/2/05	1/4/05	In Store
Mexico	1/2/05	1/2/05	In Store
Montserrat	1/2/05	1/2/05	In Store

Example 6: Creating Master Templates

PROC TEMPLATE features:

DEFINE TABLE statement:

CELLSTYLE AS statement:

STYLE variable
ROW variable

DEFINE COLUMN statement:

CELLSTYLE AS statement:

VAL variable

STYLE= column attribute statement

DEFINE HEADER statement:

STYLE= column attribute statement

LINK statement

Other ODS features: ODS HTML statement

Program Description

The following program creates four master templates for tables: Base.Template.Table, Base.Template.Column, Base.Template.Header, and Base.Template.Footer. These templates contain style information that creates alternating blue and green row colors and specific styles for headers and footers. Once they are created, master templates are applied to every table created by SAS until you specifically remove the master template or it is overridden by another table template created by PROC TEMPLATE.

Program

```
title;
options nodate nostimer LS=78 PS=60;
ods listing close;
```

Create the master parent Base.Template.Table and specify which style element and style attributes to use for different cells in a row. The DEFINE TABLE statement creates the master parent Base.Template.Table. Base.Template.Table will be applied to every table created by SAS, unless it is overridden by another template created by PROC TEMPLATE, removed with the DELETE statement, or manually removed from the item store.

The CELLSTYLE-AS statement specifies the style element and style attributes to use for cells in each of the rows in a table, which creates the alternating row colors in the output. If the row is even numbered and does not contain a style element named RowHeader, then the cell has a green background color and white font color. Similarly, if the row is even numbered and does contain a style element named RowHeader, then the cell has a blue background color and white font color.

```
proc template;
define table base.template.table;
   cellstyle mod(_row_, 2) and _style_ ^= "RowHeader" as {background=blue color=white},
             mod(_row_, 2) and _style_ = "RowHeader" as {background=green color=white};
end;
```

Create the master parent Base.Template.Column and specify which style element and style attributes to use for different cells in a column. The DEFINE TABLE statement creates the master parent Base.Template.Column. Base.Template.Column will be applied to every table created by SAS, unless it is overridden by another template created by PROC TEMPLATE, removed with the DELETE statement, or manually removed from the item store. The STYLE= column attribute statement specifies that column fonts are italicized. The first CELLSTYLE-AS statement specifies that if the value of the cell is greater than five, the font size is 15pt and if the value of the cell is equal to "Num" then the font size is 20pt.

```
define column base.template.column;
   style={fontstyle=italic};
   cellstyle _val_ > 5 as {fontsize=15pt},
             _val_ = "Num" as {fontsize=20pt};
end;
```

Create the master parent Base.Template.Header and specify the font size and font color for the headers and footers. The DEFINE TABLE statement creates the master parent Base.Template.Header. The STYLE= header attribute statement specifies that the header font is 20pt and purple. The LINK statement creates the Base.Template.Footer master template and links it to the Base.Template.Header template, which it inherits its characteristics from. Base.Template.Header and Base.Template.Footer will be applied to every table created by SAS, unless they are overridden by another template created by PROC TEMPLATE, removed with the DELETE statement, or manually removed from the item store.

```
define header base.template.header;
   style={fontsize=20pt color=purple};
end;
link base.template.footer to base.template.header;
run;
```

Create HTML output and view the contents of the SAS data set. The ODS HTML statement specifies the destination to write to and the file name of the output. The CONTENTS procedure shows the contents of the SAS data set SasHelp.Class. The ODS HTML CLOSE statement closes the HTML destination and the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

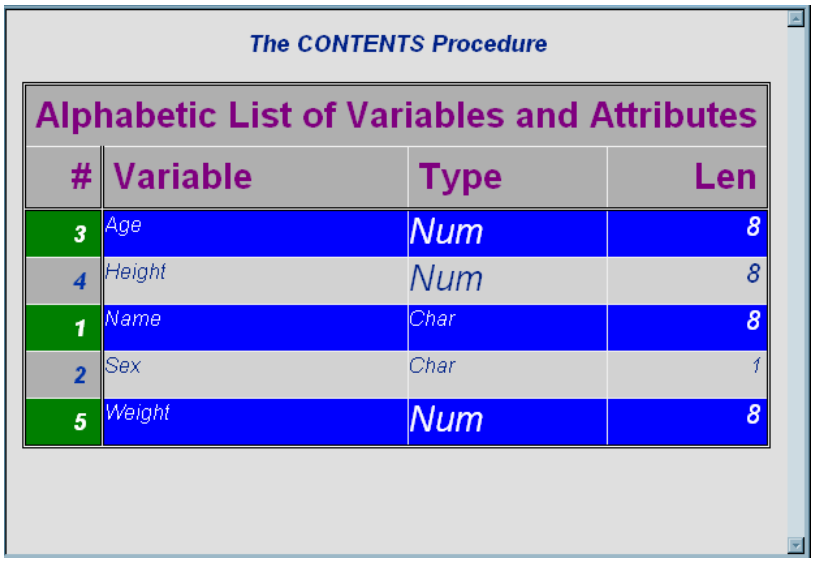
```
ods html file="MyFile.html" ;
ods select variables;
proc contents data=sashelp.class; run;

ods html close;
```

Delete the master templates. The DELETE statement deletes each master template. If you do not delete them, they will be applied to all of your tabular output until you do delete them.

```
proc template;
delete base.template.table;
delete base.template.column;
delete base.template.header;
delete base.template.footer;

run;
```

Output

Display 12.15 Using Master Templates For HTML Output



The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Example 7: Table Header and Footer Border Formatting

PROC TEMPLATE features:

Border control style attributes:

BORDERBOTTOMCOLOR=
BORDERBOTTOMSTYLE=
BORDERBOTTOMWIDTH=
BORDERTOPCOLOR=
BORDERTOPSTYLE=
BORDERTOPWIDTH=

DEFINE statement

DEFINE STYLE statement

EDIT statement

FOOTER statement

HEADER statement

PARENT= statement

PREFORMATTED= header attribute

STYLE statement

WIDTH= header attribute

Other ODS features:

ODS RTF

ODS SELECT

Data set: See “Creating the Nlits Data Set” on page 889.

Program Description

You can use the TableHeaderContainer and TableFooterContainer style elements along with the border control style attributes to change the borders of the regions surrounding the table header and footer.

Note: The TableHeaderContainer and TableFooterContainer style elements are only valid in the RTF destination. △

Program

Set the SAS system options and specify titles. The OPTIONS statement sets the SAS system options and the TITLE statements specify titles for the output.

```
options nodate nonumber;

ods listing close;

title  "TableHeaderContainer, TableFooterContainer, and Border Control Style
     Attributes";
title2 "Allows Control of Borders Between the Header, Body, and Footer of a
     Table";
```

Create the new style HeadersFootersBorders. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style HeadersFootersBorders. The PARENT= statement specifies that the new style inherits all of its style elements and style attributes from the Styles.RTF style.

```
proc template;
   define style  HeadersFootersBorders;
   parent=styles.rtf;
```

Modify the TableHeaderContainer style element. The STYLE statement with the FROM option specified creates the style element TableHeaderContainer which inherits all of its style elements and style attributes from the instance of TableHeaderContainer in the Styles.RTF style. The BORDERBOTTOMWIDTH=, BORDERBOTTOMCOLOR=, and BORDERBOTTOMSTYLE= style attributes specify the width, color, and line style of the bottom border of the table header.

```
   style TableHeaderContainer from TableHeaderContainer /
      borderbottomwidth=12
      borderbottomcolor=blue
      borderbottomstyle=dotted;
```

Modify the TableFooterContainer style element. The STYLE statement with the FROM option specified creates the style element TableFooterContainer which inherits all of its style elements and style attributes from the instance of TableFooterContainer in the Styles.RTF style. The BORDERTOPWIDTH=, BORDERTOPCOLOR=, and BORDERTOPSTYLE= style attributes specify the width, color, and line style of the top border of the table footer.

```
   style TableFooterContainer from TableFooterContainer /
      bordertopwidth=6
      bordertopcolor=red
      bordertopstyle=double;
```

Modify the Table style element. The STYLE statement with the FROM option specified creates the style element Table which inherits all of its style elements and style attributes from the instance of Table in the Styles.RTF style. The CELLSPACING=, RULES=, and FRAME= attributes modify the cellspacing, rules, and frame of the table.

```
   style table from table /
      cellspacing=0 rules=groups frame=void;
   end;
run;
```

Edit the Base.Datasets.Members table template. The EDIT statement, along with the table template DEFINE statements and attributes, modifies the Base.Datasets.Members table template. For more information about creating and modifying table templates, see Chapter 12, "TEMPLATE Procedure: Creating Tabular Output," on page 593.

```
proc template;
   edit Base.Datasets.Members;
      header hd1;
      footer ft1;
      define hd1;
         preformatted=on;
         just=l;
         text"     Table Header with Leading and Trailing Blanks    ";
      end;
      define ft1;
         preformatted=on;
         just=l;
         text"     Table Footer with Leading and Trailing Blanks    ";
      end;
      edit name;
       define header myheader;
         just=l;
         preformatted=on;
         text "     My new header";
       end;
       header=myheader;
       width=memname_width width_max=memname_width_max;
       preformatted=on;
      end;
   end;
run;
```

Create the RTF file, select the output object and run PROC DATASETS. The ODS RTF statement specifies the file that will contain the RTF output. The STYLE= option specifies the style to apply to the output. The ODS SELECT statement selects the output object Members to be sent to the open destinations.

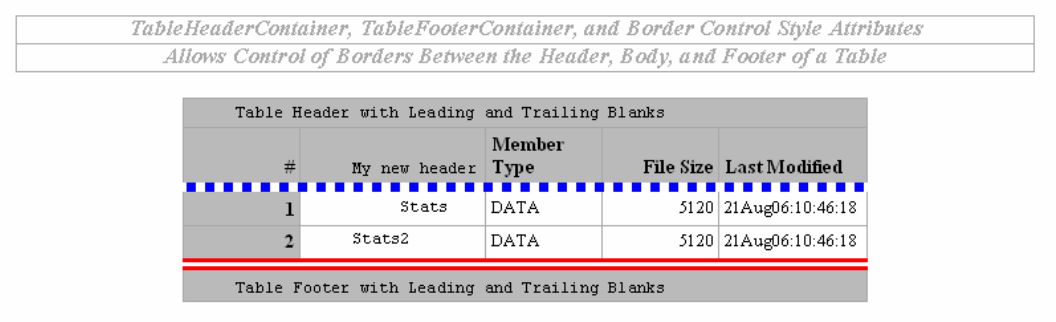
```
ods rtf file="headerfooters.rtf" style=HeadersFootersBorders;
ods select members;
proc datasets lib=nlits;
run;
quit;
```

Close the open destinations and open the LISTING destination. The ODS _ALL_ CLOSE statement closes all open destinations and the files that are associated with them. If you do not close the destinations, then you will not be able to view the files. The ODS LISTING statement opens the LISTING destination.

```
ods _all_ close;
ods listing;
```

RTF Output

Display 12.16 RTF Output with Custom Headers and Footers

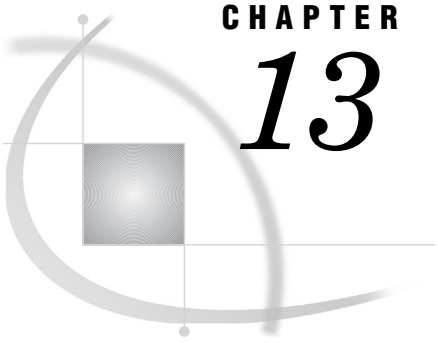


TableHeaderContainer, TableFooterContainer, and Border Control Style Attributes
Allows Control of Borders Between the Header, Body, and Footer of a Table

Table Header with Leading and Trailing Blanks

#	My new header	Member Type	File Size	Last Modified
1	Stats	DATA	5120	21Aug06:10:46:18
2	Stats2	DATA	5120	21Aug06:10:46:18

Table Footer with Leading and Trailing Blanks



CHAPTER

13

TEMPLATE Procedure: Creating Markup Language Tagsets

Overview: ODS Tagsets and the TEMPLATE PROCEDURE

The TEMPLATE procedure enables you to create a tagset, which is a type of template that defines how to generate a markup language output type from SAS output. You can specify a tagset to create markup language output from ODS. SAS provides tagsets for a variety of markup language output. For example, SAS provides several tagsets for XML output, HTML output, XSL, and more. The TEMPLATE procedure enables you to modify any of the SAS tagsets or create custom markup language tagsets.

The Output Delivery System uses the specified tagsets to mark the SAS output, which you can view with an online browser or viewer.

For information about terms used in the TEMPLATE procedure, see "Terminology: TEMPLATE Procedure" on page 402.

Markup Language Syntax: TEMPLATE Procedure

PROC TEMPLATE;

DEFINE TAGSET *tagset-path* </ STORE=*libref.template-store* >;
  <*tagset-attribute-1*; <*... tagset-attribute-n*;>>
    DEFINE EVENT *event-name*;
    <*event-attribute-1*; <*...event-attribute-n*>;>
    statements;
    END;
  NOTES;
  END;

DEFINE TAGSET Statement

Creates a tagset.

Requirement: An END statement must be the last statement in the template.

DEFINE TAGSET *tagset-path* | **Base.Template.Tagset** </
    STORE=*libref.template-store* <**(READ | WRITE | UPDATE)**>>;
  <*tagset-attribute-1*; <*... tagset-attribute-n*;>>
  DEFINE EVENT *event-name*;
statements and attributes
  NOTES *'text'*;
  END;

Table 13.1 DEFINE TAGSET Statements

Task	Statement
Define what is written to the output file	DEFINE EVENT
Provide information about the tagset	NOTES
End a tagset, or end the editing of a tagset	END

Required Arguments

tagset-path
  specifies where to store the tagset.

Requirement: A *tagset-path* consists of one or more names, which are separated by periods. Each name represents a directory, or level, in a template store.

Default: PROC TEMPLATE writes the template to the first template store in the current path where you have Write access.

Tip: Use the ODS PATH statement to control the item store where the tagset is stored.

Tip: Names are not case sensitive. However, PROC TEMPLATE puts the first letter in uppercase for easier reading.

Base.Template.Tagset
creates a tagset that is the parent of all tagsets that do not explicitly specify a parent. Once this template is created, you do not need to explicitly specify it in your SAS programs. It is automatically applied to all output until you specifically remove it from the item store.

CAUTION:
The Base.Template.Tagset supplied by SAS contains information used by many tagsets. If this information is not retained, unexpected behavior may occur. To safely create your own Base.Template.Tagset, you can start with the existing Base.Template.Tagset template by writing it to an external file and editing the existing template contents. △

Interaction: The Base.Template.Tagset master template attributes are overridden by other tagsets.

Tip: To view an existing tagset to base your own Base.Template.Tagset on, see "Viewing the Contents of a Tagset" on page 839.

Options

STORE=*libref.template-store*
specifies the template store where the template is stored in the following form:

libref.template-store <access-option(s)>

libref.template-store
specifies the current template store.

Default: If you omit an *access-option*, then the *template-store* is accessed with UPDATE permissions unless you have read-only access.

Tip: If the specified template store does not exist, then it is created.

Interaction: The STORE= option overrides the search list specified in the PATH statement.

Restriction: The STORE= option syntax does not become part of the compiled template.

access-option(s)
specifies the access mode for the specified template store.

READ
provides read-only access.

WRITE
provides Write access as well as Read access. If the tagset does not exist, then WRITE access creates a new tagset. If the tagset does exist, then WRITE access does not replace an existing tagset.

UPDATE
provides Update access as well as Read access. If the tagset does not exist, then UPDATE does not create a new tagset. If the tagset does exist, then UPDATE will replace it.

Tagset Attributes

Table 13.2 Tagset Attributes by Task

Task	Attribute
Specify the maximum number of characters that will be considered for forced line breaks by ODS	BREAKTEXT_LENGTH=
Specify the maximum ratio of the width of space available for text entry to the length of the text that is supposed to fit in that space	BREAKTEXT_RATIO=
Specify the maximum width of space available for text entry that ODS will consider for placement of automatic line breaks	BREAKTEXT_WIDTH=
Specify the text to use as a copyright	COPYRIGHT=
Specify the name of the event to use by default	DEFAULT_EVENT=
Specify whether the tagset supports embedded stylesheets	EMBEDDED_STYLESHEET=
Specify a comma-delimited list of image types or file extensions that are valid for an output destination	IMAGE_FORMATS=
Specify the number of spaces the NDENT and XDENT event statements will indent the output	INDENT=
Specify a string, which is printed to the SAS log when the tagset is used	LOG_NOTE=
Specify special characters and their translations	MAP=
Specify strings to substitute for special characters	MAPSUB=
Set a category for the output	OUTPUT_TYPE=
Specify whether a byte-order mark is written to the output files when using a UTF character set	NO_BYTE_ORDER_MARK=
Define a nonbreaking space for the markup output	NOBREAKSPACE=
Specify the tagset from which the current template inherits	PARENT=
Specify whether all style attributes are available at all times	PURE_STYLE=
Specify the text to use as a registered trademark	REGISTERED_TM=
Define a string to use for line breaks in the markup output	SPLIT=
Specify whether the tagset lets procedures place columns on top of each other, or side by side	STACKED_COLUMNS=
Specify the text to use as a trademark	TRADEMARK=

BREAKTEXT_LENGTH=*number*
specifies the maximum number of characters that will be considered for forced line breaks by ODS. When the number of characters in the text exceeds the number specified by the BREAKTEXT_LENGTH= option, then line breaks are inserted by the application that displays the output. If the number of characters in the text is less than or equal to the number specified by the BREAKTEXT_LENGTH= option, then any necessary line breaks are inserted by ODS. The placement of the line breaks is based upon the total available text width.

Example: To instruct ODS to not insert line breaks in text that is longer than 80 characters, specify the following:

```
BreakText_Length=80;
```

BREAKTEXT_RATIO=*number*
specifies the maximum ratio of the width of space available for text entry to the length of the text that is supposed to fit in that space. If the ratio of width space to text length is greater than the ratio specified by the BREAKTEXT_RATIO= option, then any necessary line breaks are inserted by the application that displays the output. If the ratio of width space to text length is equal to or less then the ratio specified by the BREAKTEXT_RATIO= option, then any necessary line breaks are inserted by ODS.

Example: To not insert line breaks into text that is more than 1.5 times longer than the width of space it is to fit in, specify the following:

```
BreakText_Ratio=1.5;
```

BREAKTEXT_WIDTH=*number*
specifies the maximum width of space available for text entry that ODS will consider for placement of automatic line breaks. If the width of space is greater than the number specified by the BREAKTEXT_WIDTH= option, then any necessary line breaks are inserted by the application that displays the output. If the width of space is less than or equal to the number specified by the BREAKTEXT_WIDTH= option, then ODS inserts necessary line breaks.

Example: To instruct ODS to not insert line breaks in text that is going into a space greater than or equal to 40 characters wide, specify the following:

```
BreakText_Width=40;
```

COPYRIGHT= '(*text*)'
specifies the text to use as the copyright.

Requirement: When specifying *text*, enclose the text in parentheses and then quotation marks.

DEFAULT_EVENT= '*event-name*'
specifies the name of an event to execute by default when the requested event cannot be found in the tagset.

Requirement: When specifying an *event-name*, enclose the name of the event in quotation marks.

Featured in: Example 3 on page 855

EMBEDDED_STYLESHEET= YES | ON | NO | OFF
specifies whether or not the tagset supports embedded stylesheets.

Default: The default value is YES or ON, which means that embedded stylesheets are supported.

Tip: If embedded stylesheets are supported and you do not specify a stylesheet in the ODS statement, then the stylesheet is written to the top of the output file.

YES
supports embedded stylesheets.
Alias: ON

ON
supports embedded stylesheets.
Alias: YES

NO
does not support embedded stylesheets.
Alias: OFF

OFF
does not support embedded stylesheets.
Alias: NO

IMAGE_FORMATS= '*image-type(s)*'
specifies a comma-delimited list of image types or file extensions that are valid for an output destination. The image types can be any that are supported by SAS/GRAPH. List them in order of preference.

Example: The following IMAGE_FORMATS= statement lists valid image types for the HTML destination:

```
image_formats='gif,jpeg,png';
```

INDENT=*n*
specifies how many spaces the NDENT and XDENT event statements will indent the output.

n
specifies a numeric value for the number of spaces that you want the output to indent.

Default: 0

Tip: The INDENT= attribute is valid only in markup family destinations.

Featured in: Example 3 on page 855 and Example 5 on page 859

LOG_NOTE= '*string*'
defines a string that is printed to the SAS log when the tagset is used.

string
specifies the text that is printed to the SAS log.

Requirement: Specify only one string at a time.

MAP= '*characters*'
specifies the special characters that require translation.

characters
specifies one or more special characters.

Requirement: When listing special characters in the MAP= statement, omit blank spaces between them.

Requirement: When you specify special characters, enclose the list of special characters in quotation marks.

Requirement: Use the MAP= statement with the MAPSUB statement.

Featured in: Example 3 on page 855

MAPSUB= '*strings*'
specifies the text to substitute for the characters that are specified in the MAP= statement.

strings
Specifies the text strings to substitute for the characters that are specified in the MAP= statement.

Requirement: When specifying multiple strings, use a forward slash (/) to separate the text strings.

Requirement: When specifying strings, enclose the entire string list in quotation marks.

Requirement: Use the MAPSUB= statement with the MAP= statement.

Featured in: Example 3 on page 855

NOBREAKSPACE= '*string*'
defines a nonbreaking space for the markup output.

string
specifies the character that defines a nonbreaking space.

Requirement: When specifying a string, enclose the string in quotation marks.

Restriction: Specify only one string at a time.

Featured in: Example 3 on page 855

NO_BYTE_ORDER_MARK=YES | ON | NO | OFF
specifies whether or not a byte-order mark is written to the output files when using a UTF character set.

OUTPUT_TYPE= CSV | HTML | LATEX | WML | XML
sets a category for the output.

CSV
produces output with comma-separated values.

HTML
produces Hypertext Markup Language output.

LATEX
produces output in LaTeX, which is a document preparation system for high-quality typesetting.

WML
uses the Wireless Application Protocol (WAP) to produce a wireless markup language.

XML
produces output in Extensible Markup Language.

Featured in: Example 3 on page 855

PARENT= *tagset-path*
specifies the tagset from which the current template inherits.

tagset-path
specifies the name of a directory in a template store.

Default: The current template inherits from the specified template in the first template store where you have Read access. The PATH statement specifies which locations to search for templates that were created by PROC TEMPLATE, as well as the order in which to search for them.

Interaction: When you specify a parent, all of the template options, attributes, and statements that are specified in the parent's template are used in the current template, unless the current template overrides them.

Requirement: When you specify a parent, all of the template options, attributes, and statements that are specified in the parent's template are used in the current template, unless the current template overrides them.

Tip: Specify a tagset that SAS supplies or a customized tagset.

Tip: Control the item store from which the tagset is read by using the ODS PATH statement.

Featured in: Example 1 on page 846 and Example 8 on page 863

PURE_STYLE=YES | ON | NO | OFF
specifies whether all of the style attributes are available at all times.

REGISTERED_TM= '(*text*)'
specifies the text to use as the registered trademark.

Requirement: When specifying *text*, enclose the text in parentheses and then quotation marks.

SPLIT= '*string*'
defines a text string to use for line breaks in the markup output.

Requirement: When specifying a string, enclose the string in quotation marks.

Restriction: Specify one string at a time.

Featured in: Example 3 on page 855

STACKED_COLUMNS= YES | ON | NO | OFF
specifies whether or not the tagset lets procedures stack columns on top of each other, or place them side by side.

Default: The default value is YES or ON, which means that columns are stacked.

Tip: To place columns side by side, specify the NO or OFF value.

Featured in: Example 3 on page 855 and Example 8 on page 863

YES
stacks columns on top of each other.

Alias: ON

ON
stacks columns on top of each other.

Alias: YES

NO
stacks columns side by side.

Alias: OFF

OFF
stacks columns side by side.

Alias: NO

TRADEMARK= '(*text*)'
specifies the text to use as the trademark.

Requirement: When specifying *text*, enclose the text in parentheses and then quotation marks.

DEFINE EVENT Statement

Defines what is written to the output file.

Interaction: You can add event statement conditions to any DEFINE EVENT statement. For more information about event statement conditions, see "Event Statement Conditions" on page 811.

Featured in: Example 6 on page 861 and Example 7 on page 863

DEFINE EVENT *event-name*;
<*event-attribute-1*;<*…event-attribute-n*;>>
BLOCK *event-name* < / *event-statement-condition(s)*>;
BREAK </ *event-statement-condition(s)*>;
CLOSE </ *event-statement-condition(s)*>;
CONTINUE </ *event-statement-condition(s)*>;
DELSTREAM $$*stream-variable-name* </ *event-statement-condition(s)*>;
DO </ *event-statement-condition(s)*>;
DONE;
ELSE </ *event-statement-condition(s)*>;
EVAL $<$>*user-defined-variable-name user-defined-variable-value where-expression* </*event-statement-condition(s)*>;
FLUSH <*/event-statement-condition(s)*>;
ITERATE $*dictionary-variable* | $*list-variable*</ *event-statement-condition(s)*>;
NDENT </ *event-statement-condition(s)*>;
NEXT $*dictionary-variable* | $*list-variable* </ *event-statement-condition(s)*>;
OPEN $$*stream-variable-name* </ *event-statement-condition(s)*>;
PUT <*function*> <NL> <*variable*> <*'text'* > < / *event-statement-condition(s)*>;
PUTL (<*variable*> | <*'text'* >| <*function*> | <NL>) < / *event-statement-condition(s)*>;
PUTLOG (<*variable*> <*'text'* > <*function*>)</ *event-statement-condition(s)*>;
PUTQ(<*variable*> | <*'text'* >| <*function*> | <NL>)</ *event-statement-condition(s)*>;
PUTSTREAM $$*stream-variable-name* </ *event-statement-condition(s)*>;
PUTVARS *variable-group variable-group-value*< / *event-statement-condition(s)*>;
SET $<$>*user-defined-variable-name user-defined-variable-value*</ *event-statement-condition(s)*>;
STOP </ *event-statement-condition(s)*>;
TRIGGER *event-name* <**START | FINISH**> </ *event-statement-condition(s)*>;
UNBLOCK *event-name* </ *event-statement-condition(s)*>;
UNSETALL | $*memory-variable* | $$*stream-variable*</ *event-statement-condition(s)*>;
XDENT </ *event-statement-condition(s)*>;
END;

Table 13.3 DEFINE EVENT Statements

Task	Statement
Add a condition to any DEFINE EVENT statement	Event statement condition(s)
Set one or more event attributes	*event-attributes*
Disable the specified event	BLOCK
Prevent an event from executing	BREAK
Close the current stream variable to which all PUT statement variables are directed	CLOSE

Task	Statement
Specify that the execution of the DO loop returns to the corresponding DO statement	CONTINUE
Delete the specified stream variable	DELSTREAM
Begin a statement block that executes if the required condition is true	DO
End a statement block	DONE
Begin a statement block that executes if the corresponding DO statement is false	ELSE
Create or update a user-defined variable and its value	EVAL
Write buffered output to the current output file or stream variable	FLUSH
Specify a dictionary variable or list variable to loop through, and for each iteration, assign the variable's values to the _NAME_ and _VALUE_ event variables	ITERATE
Indent output one more indentation level	NDENT
Increment a dictionary or list variable to the next value	NEXT
Open or create a stream variable	OPEN
Write text or variable data to an output file	PUT
Write text or variable values to an output file and add a new line to the end of the output	PUTL
Write the text or the value of a variable to the log	PUTLOG
Write text or variable values to an output file and place quotes around the value of a variable	PUTQ
Write the contents of a stream variable to the output file	PUTSTREAM
Write the name or the value of a variable to an output file	PUTVARS
Create or update a user-defined variable and its value	SET
Move execution to the end of the current statement block	STOP
Execute an event	TRIGGER
Enable a disabled event	UNBLOCK
Delete user-defined variables	UNSET
Indent output one less indentation level	XDENT
End the template	END

Required Arguments

event-name
specifies the name of the event.

event-attribute
specifies an event attribute statement. Event attribute statements are one of the following:

Table 13.4 Event Attributes

Task	Attribute	Valid Destinations
Redirect event output to any of the known types of output that are open	FILE=	HTML, MARKUP
Enable the event to use any style element that has been defined	PURE_STYLE=	MARKUP
Specify a style element	STYLE=	HTML, MARKUP

FILE= BODY | CODE | CONTENTS | DATA | FRAME | PAGES | STYLESHEET;
redirects event output to any of the known types of output files that are open.

Restriction: The FILE= attribute is valid only in markup family destinations.

Interaction: The names of the output files correspond to the output file names on the ODS MARKUP statement that are specified with the BODY=, CODE=, CONTENTS=, FRAME=, PAGES=, and STYLESHEET= options. For more information about these options, see the "ODS MARKUP Statement" on page 147.

See: The BODY= option in the "ODS MARKUP Statement" on page 147 for a complete description of the FILE= attribute

PURE_STYLE= YES | NO;
specifies whether to enable the event to use any of the style elements that have been defined.

Default: NO

Restriction: The PURE_STYLE= attribute is valid only in markup family destinations.

See also: "DEFINE STYLE Statement" on page 490

YES
enables the event to use any of the style elements that have been defined.

Alias: ON

NO
does not enable the event to use any of the style elements that have been defined.

Alias: OFF

STYLE= *style-element*;
specifies a style attribute that applies to a particular part of the output.

Restriction: The STYLE= attribute is valid only in markup family destinations.

Tip: When a carriage return separates style attributes, add a space before or after the carriage return to prevent syntax errors. SAS does not interpret a carriage return as a space.

See also: "DEFINE STYLE Statement" on page 490

Featured in: Example 6 on page 861

BLOCK Statement

Disables the specified event.

Tip: To enable the blocked event, use the UNBLOCK statement.

Tip: You can block the same event multiple times, but to enable the event, use the same number of UNBLOCK statements.

BLOCK *event* </ *event-statement-condition(s)*>;

Required Arguments

event
specifies the event.

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

BREAK Statement

Stops the current event from executing. Statements below the BREAK statement are not executed.

Tip: The BREAK statement is most useful when combined with event statement conditions.

BREAK < / *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

CLOSE Statement

Closes the current stream variable and directs all future output to the output file.

CLOSE < / *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

CONTINUE Statement

Specifies that the execution of the DO loop returns to the corresponding DO statement for re-evaluation of the IF event statement condition.

See also: "DO Statement" on page 808

CONTINUE </ *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

DELSTREAM Statement

Deletes the specified stream variable.

DELSTREAM *stream-variable* < / *event-statement-condition(s)*>;

Required Arguments

stream-variable
specifies the stream variable to be deleted.

See also: "OPEN Statement" on page 816

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (*/*).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

DO Statement

Begins a statement block that executes if the required condition is true.

DO / *event-statement-condition(s)*;

Required Arguments

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (*/*).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

DONE Statement

Ends a DO or ELSE statement block.

See also: "DO Statement" on page 808 and "ELSE Statement" on page 809

DONE;

ELSE Statement

Begins a statement block that executes if the corresponding DO statement is false.

Tip: If you specify the ELSE statement with the DO statement and the WHILE condition, then the ELSE statement executes only if the WHILE condition is false on the first evaluation.

See also: "DO Statement" on page 808

ELSE </ *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

END Statement

Ends the event.

END;

EVAL Statement

Creates or updates a user-defined variable by setting the value of the variable to the return value of a WHERE expression.

EVAL $<$>*user-defined-variable where-expression*< / *event-statement-condition(s)*>;

Required Arguments

user-defined-variable
specifies the user-defined variable that you want to create or update. A *user-defined-variable* has one of the following forms:

- $*dictionary-variable*['*key*']

- $*list-variable*[<*index*>]
- $*scalar-variable*
- $$*stream-variable*

dictionary-variable
specifies a dictionary variable to assign a *where-expression* return value. A dictionary variable is an array that contains a list of numbers or text strings that are identified by a key.

['*key*']
specifies a subscript that contains the text that identifies where in the dictionary variable you want to add the return value of the WHERE expression.

Requirement: Enclose *key* in quotation marks and brackets.

Tip: *key* is case preserving and case sensitive.

Requirement: *dictionary-variable* must be preceded by the "$" symbol.

Tip: After you create dictionary variables, they are globally available in all events until you use the UNSET statement to delete them.

See also: Dictionary variable on page 841 for more information

list-variable
specifies a list variable to which to assign a *where-expression* return value. A list variable is an array that contains a list of numbers or text strings that are indexed.

[<*index*>]
specifies a subscript that contains a number or numeric variable.

The *index* identifies the location in the list to add the return value of the WHERE expression. If you omit the *index* and specify only empty brackets, or if the value of *index* is greater than the highest *index* number, then the EVAL statement appends the return value to the end of the list.

Requirement: Specify brackets [ ], even if you omit an index.

Requirement: Enclose *index* in brackets.

See also: List variable on page 841 for more information

Requirement: *list-variable* must be preceded by a '$' symbol.

Tip: List variables are accessed sequentially by using the ITERATE and NEXT statements.

Tip: After you create list variables, they are globally available in all events until you use the UNSET statement to delete them.

scalar variable
specifies a scalar variable to which to assign a *where-expression* return value.

Requirement: Scalar variables must be preceded by the '$' symbol.

Requirement: After you create scalar variables, they are globally available in all events and persist until you use the UNSET statement to unset them.

stream-variable
specifies a stream variable to which you want to assign a *where-expression* return value. A stream variable is a temporary item store that contains output.

While the stream variable is open, all output from PUT statements is directed to the stream variable until it is closed.

Requirement: *stream-variable* must be preceded by the "$$" symbol.

See also: "Understanding Variables" on page 840 for information about stream variables

where-expression
any expression that can be used in the WHERE= data set option.

See: For information on expressions that you can use in the WHERE data set option, see the WHERE= data set option in *SAS Language Reference: Dictionary* and "WHERE Expression Processing" in *SAS Language Reference: Concepts*.

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

Event Statement Conditions

Specify one or more conditions that must be true for a DEFINE EVENT statement to execute.

Requirement: An event statement condition must be preceded with a slash (/).

---

define-event-statement </ *event-statement-condition(s)*>;

Event Statement Conditions

define-event-statement
specifies a DEFINE EVENT statement.

event-statement-condition
specifies a condition to evaluate.
event-statement-condition is one of the following:

ANY (*variable-1*,< …, *variable-n*>)
checks a list of comma-delimited variables for values. If any of the variables has a value, then the condition is true.

Example:

```
put 'One of our variables has a value!'
   nl/if any(background, foreground, cellpadding, cellspacing);
```

BREAKIF *event*
stops an event that is executing. The current statement is executed and the event ends.

Tip: Using the BREAKIF condition is more efficient than using a PUT event statement and a BREAK event statement with an IF condition together. For example, the following statements are equivalent:

```
put 'Foreground has a value!' /breakif exists(foreground);
```

```
put 'Foreground has a value!' /if exists(foreground);
break /if exists(foreground);
```

CMP (“*string*”, *variable* | *variable-list*)
compares, for equality, a string to a variable or list of variables.

Example:

```
put 'The foreground is blue!' nl/if cmp('blue',foreground);
```

CONTAINS (*argument-1*, *argument-2*)
searches the first argument for the second argument.

Example:

```
set $myvariable 'some random text';
put 'myvariable contains 'ran' nl/if contains($myvariable, 'ran');
```

EXIST | EXISTS (*variable* | *variable-list*)
determines whether a variable or a list of variables has values. If all of the variables have values, then the condition is true. If a variable has an empty string of length 0, then the variable has no value and the condition is false.

Tip: Use the MISSING event variable with the EXIST condition to determine whether a value is missing.

Example:

```
put 'All of our variables have a value!'
   nl/if exists(background, foreground, cellpadding, cellspacing);
```

IF | WHEN | WHERE(<*value*><*'string'*><*variable*>)
tests for existence or equality. IF, WHEN, and WHERE are optional and interchangeable. An IF, a WHEN, or a WHERE condition compares values and strings, or checks variables for values.

Example: All of the following are equivalent:

```
put 'Foreground has a value!' nl/if (foreground);
put 'Foreground has a value!' nl/if exists(foreground);
put 'Foreground has a value!' nl/when exists(foreground);
put 'Foreground has a value!' nl/exists(foreground);
put 'Foreground has a value!' nl/where existsforeground);
```

Restriction: When you specify an IF condition with a single, user-defined variable, then the variable is evaluated to determine if it has a value, according to the variable's type.

Table 13.5 Variable Type and Criteria

Variable Type	Criteria to Determine Existence
String	Length
Numeric	Value
Dictionary array	Key (if there is a key specified, then the test is true)

NOT | ! | ^ <*'string'*><*variable*>
negates a condition. You can use the keyword NOT or the characters '!' or '^'.

Restriction: The character '!' works only as the first character in a condition. The standard WHERE processing syntax is required for subsequent characters.

Example:

```
put 'The foreground is not red!' nl/if not cmp('red', foreground);
put 'The foreground is not red or blue' /if !cmp('red', foreground)
     and ^cmp('blue', foreground);
put 'The foreground is not red or blue' /if ^cmp('red', foreground)
     and ^cmp('blue', foreground);
```

WHILE *condition-expression*
indicates that the corresponding statement block should loop until the WHILE value becomes false.

Restriction: The WHILE condition can be used only with the DO statement.

Example:

```
eval $count 0;

do /while $count < 10;
    eval $i $count+1;
    continue /if $count eq 5;
    stop /if $count eq 8;
    put 'Count is ' $i nl;
else;
    put 'Count was never less than 10' nl;
done;
```

FLUSH Statement

Writes buffered output to the current output file or the current stream variable.

FLUSH </ *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

ITERATE Statement

Specifies a dictionary variable or list variable to loop through, and assigns the variable's value to the _NAME_ and _VALUE_ event variables for each iteration.

Requirement: You must use the ITERATE statement with the _VALUE_ or _NAME_ event variables. The first value of the dictionary variable or list variable is placed in the _VALUE_ event variable. For dictionary variables, the *key* is placed in the _NAME_ event variable.

See also: _VALUE_ on page 834 and _NAME_ on page 834

ITERATE *dictionary-variable | list-variable </ event-statement-condition(s)>*;

Required Arguments

dictionary-variable
specifies a dictionary variable.

Requirement: *dictionary-variable* must be preceded by the "$" symbol.

Tip: User-defined variables are not case sensitive.

See also: The "EVAL Statement" on page 809 or the "SET Statement" on page 825 for information about dictionary variables

list-variable
specifies a list variable.

Requirement: *list-variable* must be preceded by the "$" symbol.

Tip: User-defined variables are not case sensitive.

See also: The "EVAL Statement" on page 809 or the "SET Statement" on page 825 for information about list variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (*/*).

See: "Event Statement Conditions" on page 811 for information about these conditions

NDENT Statement

Indents output one more level than the number of spaces specified by the INDENT= attribute.

Interaction: The start position of the indention level is set by the INDENT= attribute.

Featured in: Example 3 on page 855 and Example 5 on page 859

NDENT < / *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

NEXT Statement

Specifies to increase a dictionary or list variable incrementally to the next value and to repopulate the event variables _VALUE_ and _NAME_ as appropriate.

Requirement: Use the NEXT statement with the ITERATE statement.

See also: _VALUE_ on page 834 and _NAME_ on page 834

NEXT $*dictionary-variable* | $*list-variable* </ *event-statement-condition(s)*>;

Required Arguments

dictionary-variable
specifies a dictionary variable that is designated as an iterator by the ITERATE statement.

Requirement: *dictionary-variable* must be preceded by the "$" symbol.

Tip: User-defined variables are not case sensitive.

See also: "ITERATE Statement" on page 814

See also: The "EVAL Statement" on page 809 or the "SET Statement" on page 825 for information about dictionary variables

list-variable
specifies a list variable that is designated as an iterator by the ITERATE statement.

Requirement: *list-variable* must be preceded by the "$" symbol.

Tip: User-defined variables are not case sensitive.

See also: "ITERATE Statement" on page 814

See also: The "EVAL Statement" on page 809 or the "SET Statement" on page 825 for information about list variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (*/*).

See: "Event Statement Conditions" on page 811 for information about these conditions

OPEN Statement

Opens or creates a stream variable. When the PUT statements occur after the OPEN statement, all text or variable data that is specified by PUT statements is appended to the stream variable instead of the output file.

Interaction: An open stream variable is closed when a new stream variable is opened.

OPEN *stream-variable* </ *event-statement-condition(s)*>;

Required Arguments

stream–variable
specifies a stream variable, which is a temporary item store that contains output.

Tip: User-defined variables are not case sensitive.

Tip: If you assign the name of a memory variable to *stream-variable*, then the stream variable resolves as the value of the memory variable. For example, the following program uses the memory variable $MyStream as a stream variable:

```
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is being used as a stream variable';
close;
```

Therefore, the following statements are equivalent:

```
put $$test;
putstream $mystream;
putstream test;
```

The following statements are also equivalent:

```
unset $$test;
delstream $mystream;
delstream test;
```

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

PUT Statement

Writes text, new lines, variable values, or DATA step function return values to an output file.

Featured in: Example 1 on page 846, Example 3 on page 855, Example 4 on page 857, Example 5 on page 859, and Example 6 on page 861

PUT <'*text*'> <NL(s)> <*value(s)*> </ *event-statement-condition(s)*>;

Required Arguments

NL
specifies a new line.

Alias: CR

Alias: LF

text
specifies a string of text.

Requirement: *text* must be enclosed in quotation marks.

Interaction: The PUT statement pairs text strings with variables. A string of text that precedes a variable creates a string-value pair if the variable has a value. For example, for the following PUT statement, if the event variable ForeGround has a value of **`blue`**, then the output is **`color=blue`**:

```
put 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUT statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output is **`<table>`** followed by a new line:

```
put '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>' nl;
```

value
specifies any event variable, style variable, dynamic variable, user-defined variable, or DATA step function whose value you want to output.

Restriction: DATA step functions cannot be nested.

Requirement: User-defined variables must be preceded by a '$' or '$$' character.

Interaction: The PUT statement pairs text strings with variables. A string of text that precedes a variable creates a string-value pair, if the variable has a value. For example, for the following PUT statement, if the event variable ForeGround has a value of **`blue`**, then the output is **`color=blue`**:

```
put 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUT statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output is **<table>** followed by a new line:

```
put '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>' nl;
```

Tip: User-defined variables are not case sensitive.

See also: *SAS Language Reference: Dictionary* for information about DATA step functions

See also: "Understanding Variables" on page 840 for information about variables

See also: "Event Variables" on page 833 for a list of event variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (*/*).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

PUTL Statement

Writes text, new lines, variable values, or DATA step function return values to an output file and automatically adds a new line to the end of the output.

Tip: When the output is large, it is useful to use the PUTL statement because it adds a new line to the end of the output.

PUTL <*'text'*> <NL(s)> <*value(s)*> </ *event-statement-condition(s)*>;

Required Arguments

NL
specifies a new line.

Alias: CR

Alias: LF

text
specifies a string of text.

Requirement: *text* must be enclosed in quotation marks.

Interaction: The PUTL statement pairs text strings with variables. A string of text that precedes a variable creates a string-value pair if the variable has a value. For example, for the following PUTL statement, if the event variable ForeGround has a value of **blue**, then the output is **color=blue** followed by a new line:

```
putl 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUTL statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output is **<table>** followed by two new lines:

```
putl '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>' nl;
```

value
specifies any event variable, style variable, dynamic variable, user-defined variable, or DATA step function whose value you want to output.

Restriction: DATA step functions cannot be nested.

Requirement: User-defined variables must be preceded by a '$' or '$$' character.

Interaction: The PUTL statement pairs strings with variables. A string of text that precedes a variable creates a string-value pair if the variable has a value. For example, for the following PUTL statement, if the event variable ForeGround has a value of **blue**, then the output is **color=blue**, followed by a new line:

```
putl 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUTL statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output is **<table>** followed by two new lines: one that is specified and the other that is generated automatically:

```
putl '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>' nl;
```

Tip: User-defined variables are not case sensitive.

See also: *SAS Language Reference: Dictionary* for information about DATA step functions

See also: "Understanding Variables" on page 840 for information about variables

See also: "Event Variables" on page 833 for a list of event variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

PUTLOG Statement

Writes text, new lines, variable values, or DATA step function return values to the log.

Restriction: Unlike the other PUT statements, the PUTLOG statement does not specify new lines.

PUTLOG <*'text'*> <*value(s)*> </ *event-statement-condition(s)*>;

Required Arguments

text
specifies a string of text.

Requirement: *text* must be enclosed in quotation marks.

Interaction: The PUTLOG statement pairs text strings with variables. A string of text that precedes a variable creates a string-value pair if the variable has a value. For example, for the following PUTLOG statement, if the event variable ForeGround has a value of **blue**, then the output that is written to the log is **color=blue**:

```
putlog 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUT statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output that is written to the log is **<table>**:

```
putlog '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>';
```

value
specifies any event variable, style variable, dynamic variable, user-defined variable, or DATA step function whose value you want to output.

Restriction: DATA step functions cannot be nested.

Requirement: User-defined variables must be preceded by a '$' or '$$' character.

Interaction: The PUTLOG statement pairs text strings with variables. A string of text that precedes a variable creates a string-value pair, if the variable has a value. For example, for the following PUTLOG statement, if the event variable ForeGround has a value of **blue**, then the output that is written to the log is **color=blue**:

```
putlog 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUTLOG statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output that is written to the log is **<table>**:

```
putlog '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>';
```

Tip: User-defined variables are not case sensitive.

See also: *SAS Language Reference: Dictionary* for information about DATA step functions

See also: "Understanding Variables" on page 840 for information about variables

See also: "Event Variables" on page 833 for a list of event variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

PUTQ Statement

Writes text, new lines, variable values, or DATA step function return values to an output file and places quotes around the value of the variable.

Featured in: Example 7 on page 863

PUTQ <*'text'*> <NL(s)> <*value(s)*> </ *event-statement-condition(s)*>;

Required Arguments

NL
specifies a new line.

Alias: CR

Alias: LF

text
specifies a string of text.

Requirement: *text* must be enclosed in quotation marks.

Interaction: The PUTQ statement pairs strings with variables. A string of text that precedes a variable creates a string-value pair if the variable has a value. For example, for the following PUTQ statement, if the event variable ForeGround has a value of **blue**, then the output is **color='blue'**:

```
putq 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUTQ statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output is **<table>**, followed by a new line:

```
putq '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>' nl;
```

value
specifies any event variable, style variable, dynamic variable, user-defined variable, or DATA step function whose value you want to output.

Restriction: DATA step functions cannot be nested.

Requirement: User-defined variables must be preceded by a '$' or '$$' character.

Interaction: The PUTQ statement pairs text strings with variables. A string of text that precedes a variable creates a string-value pair, if the variable has a value. For example, for the following PUTQ statement, if the event variable ForeGround has a value of **blue**, then the output is **color=blue**:

```
putq 'color=' foreground;
```

If the variable does not have a value, then the text is not written, and there is no output for the text or the variable. For example, for the following PUTQ statement, if the variables BackGround, ForeGround, and CellPadding do not have values, then the output is **<table>**, followed by a new line:

```
putq '<table' 'background=' background 'foreground=' foreground
    'cellpadding=' cellpadding '>' nl;
```

Tip: User-defined variables are not case sensitive.

See also: *SAS Language Reference: Dictionary* for information about DATA step functions

See also: "Understanding Variables" on page 840 for information about variables

See also: "Event Variables" on page 833 for a list of event variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

PUTSTREAM Statement

Writes the contents of the specified stream variable to an output file.

PUTSTREAM *stream-variable* < / *event-statement-condition(s)*>;

Required Arguments

stream-variable
specifies a stream variable, which is a temporary item store that contains output.

Tip: If you assign the name of a memory variable to *stream-variable-name*, then the stream variable resolves as the value of the memory variable. For example, the following partial program uses the memory variable $MyStream as a stream variable:

```
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is being used as a stream variable';
close;
```

Therefore, the following statements are equivalent:

```
put $$test;
putstream $mystream;
putstream test;
```

The following statements are also equivalent:

```
unset $$test;
delstream $mystream;
delstream test;
```

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

PUTVARS Statement

Iterates over each value in a variable group, list, or dictionary and writes text, new lines, variable values, or DATA step function return values to an output file. Each iteration populates the special variables _VALUE_ and __NAME_ . Putvars prints once for each variable or value that it finds.

Tip: The variable _NAME_ contains the name of the variable. The variable _VALUE_ contains the value of the variable.

See also: _VALUE_ and _NAME_ in the "Tables of Event Variables" on page 833

PUTVARS (*variable-group* | *dictionary-variable* | *list-variable*) <NL(s)> <*'text'*>
<*value(s)* >
< / *event-statement-condition(s)*>;

Required Argument

variable-group
specifies the type of variable to use in each iteration when you specify the name or value in the variable. For example, if you specify the EVENT option, then the

PUTVARS statement loops through all of the event variables in the program. *variable-group* is one of the following:

EVENT
specifies to loop through all event variables.

See also: "Event Variables" on page 833

STYLE
specifies to loop through all style variables.

DYNAMIC
specifies to loop through all dynamic variables.

MEMORY
specifies to loop through all memory variables. A memory variable is classified as a dictionary variable if it is created with a subscript that contains a key. A memory variable is classified as a list variable if it is created with a subscript that is empty or contains an index. If you omit a key or an index, then the memory variable is a numeric or character scalar variable, depending on the variable's value.

Restriction: The PUTVARS statement ignores list or dictionary memory variables.

STREAM
specifies to loop through all stream variables.

Interaction: The PUTVAR statement pairs text strings with variables. If a string is followed by a variable, then they become a pair. If the variable has a value, then the pair becomes output. If the variable does not have a value, then neither becomes output.

dictionary-variable
specifies a dictionary variable.

Requirement: *dictionary-variable* must be preceded by the '$' symbol.

Tip: User-defined variables are not case sensitive.

See also: For information about list variables, see the following sections:

- "EVAL Statement" on page 809
- "SET Statement" on page 825
- "Understanding Variables" on page 840

list-variable
specifies a list variable.

Requirement: *list-variable* must be preceded by the "$" symbol.

Tip: User-defined variables are not case sensitive.

See also: For information about list variables, see the following sections:

- "EVAL Statement" on page 809
- "SET Statement" on page 825
- "Understanding Variables" on page 840

NL
specifies a new line.

Alias: CR

Alias: LF

text
specifies a string of text.

Requirement: *text* must be enclosed in quotation marks.

value

specifies any event variable, style variable, dynamic variable, user-defined variable, or DATA step function whose value you want to output.

Restriction: DATA step functions cannot be nested.

Requirement: User-defined variables must be preceded by a '$' or '$$' character.

Tip: User-defined variables are not case sensitive.

See also: *SAS Language Reference: Dictionary* for information about DATA step functions

See also: "Understanding Variables" on page 840 for information about variables

See also: "Event Variables" on page 833 for a list of event variables

Options

event-statement-condition(s)

specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

SET Statement

Creates or updates a user-defined variable and its value.

❶ **SET** $*dictionary-variable entry* </ *event-statement-condition(s)*>;

❷ **SET** $*list-variable entry* </ *event-statement-condition(s)*>;

❸ **SET** $*scalar-variable* | $$*stream-variable entry* </ *event-statement-condition(s)*>;

Required Arguments

dictionary-variable

specifies an array that contains a list of numbers or text strings that is identified by a key. *dictionary-variable* has the following form:

$*dictionary-variable*['*key*']

['*key*']

specifies a subscript that contains text or a variable that has a character value.

Requirement: Enclose *key* in quotation marks and brackets.

Tip: *key* is case preserving and case sensitive.

Example: The following example puts two key value pairs into the dictionary variable MyDictionary:

```
set $mydictionary['URL1'] 'links internally';
set $mydictionary['URL2'] 'links externally';
```

Requirement: *dictionary-variable* must be preceded by the '$' symbol.

Tip: Dictionary variables are accessed sequentially by using the ITERATE and NEXT statements.

Tip: After they are created, dictionary variables are globally available in all events until you delete them by using the UNSET statement.

entry
specifies the value of a dictionary variable, list variable, scalar variable, or stream-variable. An *entry* is one of the following:

function
specifies a DATA step function.

Restriction: Functions cannot be nested.

See also: *SAS Language Reference: Dictionary* for information on SAS functions

text
specifies a string of text.

Requirement: *text* must be enclosed in quotation marks.

variable
specifies any event variable, style variable, dynamic variable, user-defined variable, or DATA step function whose value you want to output.

Restriction: *variable* cannot be a stream variable.

Requirement: User-defined variables must be preceded by a '$' character.

Tip: If you assign a *variable* entry that is the name of a memory variable to *stream-variable*, then the stream variable resolves as the value of the memory variable. For example, the following program uses the memory variable $MyStream as a stream variable:

```
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is being used as a stream variable';
close;
```

Therefore, the following statements are equivalent:

```
put $$test;
putstream $mystream;
putstream test;
```

The following statements are also equivalent:

```
unset $$test;
delstream $mystream;
delstream test;
```

Tip: User-defined variables are not case sensitive.

See also: "Understanding Variables" on page 840 for information about variables

See also: "Event Variables" on page 833 for a list of event variables

list-variable
an array that contains a list of numbers or strings of text that are indexed. *list-variable* has the following form:

$*list-variable*[<*index*>]

[<*index*>]
specifies a subscript that contains a number or numeric variable. The index identifies the location in the list to add an *entry*. If you omit the index and only

specify empty brackets, or if the value of the index is greater than the highest index number, then the SET statement appends the *entry* to the end of the list.

Requirement: Specify brackets [ ], even if you omit an index.

Requirement: Enclose *index* in brackets.

Tip: List entries are accessed by positive or negative indexes. Positive indexes start at the beginning of a list. Negative indexes start at the end of a list. For example, the following list variable, $Mylist[2], identifies the second entry in the list variable $Mylist. In this case, the index is 2. The list variable $Mylist[-2] identifies the second entry from the end of the list variable $Mylist. In this case, the index is [-2].

Example: The following example adds three values onto the end of the list variable MyList and modifies the value of the second entry.

```
set $mylist[] 'one';
set $mylist[] 'two';
set $mylist[] 'hello';
set $mylist[2] 'This is Really two';
```

Requirement: *list-variable* must be preceded by a '$' symbol.

Tip: List variables are accessed sequentially by using the ITERATE and NEXT statements.

Tip: After they are created, list variables are globally available in all events until you delete them using the UNSET statement.

scalar-variable

an area of memory that contains numeric or character data.

Requirement: Scalar variables must be preceded by the '$' symbol.

Tip: After they are created, list variables are globally available in all events until you delete them using the UNSET statement.

stream-variable

specifies a stream variable, which is a temporary item store that contains output. While the stream variable is open, all output from PUT statements is directed to the stream variable until it is closed.

Requirement: *user-defined-variable-name* must be preceded by the '$$' symbol.

Tip: If you assign a variable entry that is the name of a memory variable to *stream-variable*, then the stream variable resolves as the value of the memory variable. For example, the following program uses the memory variable $MyStream as a stream variable:

```
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is being used as a stream variable';
close;
```

Therefore, these statements are equivalent:

```
put $$test;
putstream $mystream;
putstream test;
```

These statements are also equivalent:

```
unset $$test;
delstream $mystream;
delstream test;
```

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (*/*).

See: "Event Statement Conditions" on page 811 for information about these conditions

❶ Adding Entries to Dictionary Variables

Use this form of the SET statement to add an entry to a dictionary variable.

SET $*dictionary-variable entry* </ *event-statement-condition(s)*>;

A dictionary variable is an array that contains a list of numbers or text strings that is identified by a key. A dictionary variable has, as part of its name, a preceding '$' symbol and a subscript that contains a text string or a variable that has a character value. The text or variable within the subscript is called a key. Keys are case preserving and case sensitive. After they are created, dictionary variables are globally available in all events and persist until you unset them with the UNSET statement.

An entry is a variable, string of text, or function. If a string of text follows the dictionary variable, then the entry becomes a key-value pair. For example, the following program adds two key-value pairs to a dictionary:

```
set $mydictionary['URL1'] 'links internally';
set $mydictionary['URL2'] 'links externally';
```

❷ Adding Entries to List Variables

Use this form of the SET statement to add an entry to a list variable.

SET $*list-variable entry* </ *event-statement-condition(s)*>;

A list variable is an array that contains a list of numbers or text strings that are indexed. As part of their name, list variables have a preceding '$' symbol and a subscript that is empty or contains a number or numeric variable. The number within the subscript is called an index. After they are created, list variables are globally available in all events and persist until you unset them with the UNSET statement. List entries are accessed by positive or negative indexes. Positive indexes start at the beginning of a list. Negative indexes start at the end of a list.

For example, the following list variable, $Mylist[2], identifies the second entry in the list variable $Mylist. In this case, the index is 2. The list variable $Mylist[-2] identifies the second entry from the end of the list variable $Mylist. In this case, the index is [-2].

STOP Statement

Specifies that execution moves to the end of the current statement block.

STOP </ *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition(s)* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

---

TRIGGER Statement

Executes an event.

Tip: The TRIGGER statement explicitly requests a specific action of an event.

Featured in: Example 3 on page 855, Example 4 on page 857, Example 5 on page 859, and Example 6 on page 861

TRIGGER *event-name* <**START | FINISH**> </ *event-statement-condition(s)*>;

Required Arguments

event-name
specifies the name of the event.

Without Options

If a triggered event does not have start or finish sections, then it runs the current event statements.

Options

START
specifies the start section of an event.

Interaction: If the program is in the start section of an event, then any event that is triggered runs its start section.

FINISH
specifies the finish section of an event.

Interaction: If the program is in the finish section of an event, then any event that is triggered runs its finish section.

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: an *event-statement-condition* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

UNBLOCK Statement

Enables a disabled event.

Interaction: To disable an event, use the BLOCK statement.

Requirement: Because you can block the same event multiple times, to enable the event use the same number of UNBLOCK statements as BLOCK statements.

UNBLOCK *event-name* </ *event-statement-condition(s)*>;

Required Arguments

event-name
specifies the name of the event.

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: an *event-statement-condition* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

UNSET Statement

Deletes a user-defined variable and its value.

UNSET ALL | *dictionary-variable* | *list-variable* | *scalar-variable* | *stream-variable* </ *event-statement-condition(s)*>;

Required Arguments

ALL
deletes all dictionary variables, list variables, and scalar variables.

Tip: You must delete stream variables individually.

dictionary-variable
specifies an array that contains a list of numbers or text strings that are identified by a key. A *dictionary-variable* has the following form:

$*dictionary-variable*['*key*']

['*key*']
specifies the location in the dictionary variable of the value that you want to delete.

Requirement: Enclose *key* in quotation marks and brackets.

Requirement: *key* must be a string of text or a character variable.

Tip: *key* is case preserving and case sensitive.

Requirement: A *dictionary-variable* must be preceded by the '$' symbol.

list-variable
specifies an array that contains a list of numbers or strings of text that are indexed. A *list-variable* has the following form:

$*list-variable*[<*index*>]

[<*index*>]
specifies the location in the list variable of the value to be deleted. If you omit the *index* and specify empty brackets, then the entire list variable is deleted.

Requirement: Specify brackets [ ], even if you omit an index.

Requirement: *index* must be number or numeric variable.

Requirement: Enclose *index* in brackets.

Tip: List entries are accessed by positive or negative indexes. Positive indexes start at the beginning of a list. Negative indexes start at the end of a list. For example, in the following code, the first UNSET statement deletes the first entry from the top of the list variable MyList. The second UNSET statement deletes the first entry from the bottom of the MyList list variable:

```
unset $mylist[-1];
unset $mylist[1];
```

Requirement: A *list-variable* must be preceded by a '$' symbol.

scalar-variable
specifies a scalar variable to delete.

Requirement: Scalar variables must be preceded by the '$' symbol.

See also: "SET Statement" on page 825 or "Understanding Variables" on page 840 for information on scalar variables

stream-variable
specifies a stream variable to delete.

Requirement: A *user-defined-variable-name* must be preceded by the '$$' symbol.

Tip: If you assign a variable entry that is the name of a memory variable to *stream-variable*, then the stream variable resolves as the value of the memory variable. For example, the following program uses the memory variable $MyStream as a stream variable:

```
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is being used as a stream variable';
close;
```

Therefore, the following statements are equivalent:

```
put $$test;
putstream $mystream;
putstream test;
```

The following statements are also equivalent:

```
unset $$test;
delstream $mystream;
delstream test;
```

See also: "SET Statement" on page 825 or "Understanding Variables" on page 840 for information on memory variables

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: An *event-statement-condition* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

XDENT Statement

Indents output one less indention level, using the number of spaces specified by the INDENT= attribute.

Interaction: The starting level of indention is set by the NDENT= statement.

Featured in: Example 3 on page 855 and Example 5 on page 859

XDENT </ *event-statement-condition(s)*>;

Options

event-statement-condition(s)
specifies one or more conditions that must be true for the event statement to execute.

Requirement: *event-statement-condition* must be preceded by a slash (/).

See: "Event Statement Conditions" on page 811 for information about these conditions

NOTES Statement

Provides information about the tagset.

Tip: The NOTES statement becomes part of the compiled tagset, which you can view with the SOURCE statement.

Featured in: Example 3 on page 855 and Example 8 on page 863

NOTES *'text'*;

Required Arguments

text
provides information about the tagset.

Requirement: When specifying *text*, enclose the text in quotation marks.

END Statement

Ends the tagset.

END;

Event Variables

Event variables include text, formatting, and data values that are associated with events. These variables originate in many places, such as table templates, the procedures, titles, bylines, and processing. Event variables also include any style attributes that are used in the program. The following table lists the internally generated event variables that are used in the DEFINE EVENT statement of PROC TEMPLATE.

Tables of Event Variables

Table 13.6 508 Accessibility* Variables

Event Variable	Description
ABBR	Specifies an abbreviation for the event variable.
ACRONYM	Specifies an acronym for an event variable.
ALT	Specifies an alternate description of an event variable.
CAPTION	Specifies the caption for a table.
LONGDESC	Specifies the long description of an event variable.
SUMMARY	Specifies a summary of a table.

* SAS includes these accessibility and compatibility features to improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology that are adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

Table 13.7 Data Variables

Event Variable	Description
NAME	Contains the name of the current variable.
VALUE	Contains the value of the current variable.
DNAME	Specifies the name of the column in the data component to associate with the current column. DNAME is specified with the DATANAME= attribute in a column template. For information, see the DATANAME= attribute on page 605.
LABEL	Specifies a label for the variable. The LABEL event variable is set with the LABEL= attribute in the column template. For information, see the LABEL= attribute on page 609.
NAME	Specifies the name of the variable. NAME is set with the VARNAME= attribute in the column template. For information, see the VARNAME= attribute on page 613.
VALUE	Specifies the current value.
VALUECOUNT	Specifies the count of the variable.

Table 13.8 Event Meta Variables

Event Meta Variables	Description
EMPTY	Sets a flag to determine whether an event is called as an empty tag.
EVENT_NAME	Specifies the requested event name.
STATE	Specifies the current state of the event, which is either START or FINISH.
TRIGGER_NAME	Specifies the name of the event that is triggered.

Table 13.9 Formatting Data

Event Variable	Description
CLOSURE	Specifies whether the endpoints of a format range are included or excluded, for example (<-, -, -<, <-<, and so on).
COL_ID	Specifies the column ID to identify columns. Used for the OIMDBM format type by the XML LIBNAME engine.
DATAENCODING	Specifies the encoding type for Raw value. It is always Base64.

Event Variable	Description
MISSING	Specifies the value that indicates that no data value is stored. By default, SAS uses a single period (.) for a missing numeric value and a blank space for a missing character value. In addition, for a numeric missing value, a special missing value indicator represents different categories of missing data by assigning one of the letters A through Z, or an underscore.
NO_WRAP	Specifies that the current cell should not wrap text or insert hyphens.
PRECISION	Specifies the number of places to the right of the decimal. The PRECISION variable is used by the XML LIBNAME engine.
RANGEEND	Specifies the end value of a range in a format.
RANGESTART	Specifies the start value of a range in a format
RAWVALUE	Specifies the base64 encoding of the stored machine representation of the original value.
SASFORMAT	Specifies the SAS format used to format a value.
SCALE	Specifies the total number of places in the floating point number. The SCALE event variable is used by the XML LIBNAME engine.
TYPE	Specifies the STRING, DOUBLE, CHAR, BOOL, or INT data type.
UNFORMATTEDTYPE	Specifies the data type before formatting.
UNFORMATTEDVALUE	Specifies the value before formatting.
UNFORMATTEDWIDTH	Specifies the width before formatting.

Table 13.10 General Use Variables

Variable	Description
ANCHOR	Specifies the current anchor, which is the last value of the anchor tag (for example, IDX).
DATA_VIEWER	Specifies the name of the Data Viewer, such as Table, Batch, Tree, Graph, Report, or Print.
DATE	Specifies the date.
DEST_FILE	Specifies the current destination file, which is one of the following: body, contents, pages, frame, code, or stylesheet.
FIRSTPAGE	Specifies the first page of the output file.
LANGUAGE	Specifies the language of the current output. The LANGUAGE event variable is set only when it is an Asian language.
OUTPUT_LABEL	Specifies the label of the current output object.

Variable	Description
OUTPUT_NAME	Specifies the name of the current output object.
OUTPUT_TYPE	Specifies the output type as specified in the tagset.
PAGE_COUNT	Specifies the page count since the files were opened.
PROC_COUNT	Specifies how many procedures have run since the files were opened.
PROC_NAME	Specifies the name of the current procedure.
SASLONGVERSION	Specifies the long format of the SAS version.
SASVERSION	Specifies the short format of the SAS version.
SPACE	Specifies the string that the tagset uses for a nonbreaking space.
SPLIT	Specifies the string that the tagset uses for line breaks.
STYLE	Specifies the current style that is in use.
STYLE_ELEMENT	Specifies the name of the current style element.
SUPPRESS_CHARSET	Specifies the Suppress Charset Registry setting.
TIME	Specifies the time.
TOCLEVEL	Specifies the table of contents level.
TOTAL_PAGE_COUNT	Specifies the total page count since ODS was opened.
TOTAL_PROC_COUNT	Specifies the number of procedures that have run since ODS was opened.

Table 13.11 ODS Statement Variables: Variables That Originate with the ODS Statement That Invoked the Tagset

Event Variable	Description
AUTHOR	Specifies the author of the output. The value of the AUTHOR event variable is set from an ODS statement, or, by default, is the user that is running SAS.
BASENAME	Specifies the name of the BASE= option as set in an ODS statement.
BODY_NAME	Specifies the name of the body file.
BODY_TITLE	Specifies the title of the body file.
BODY_URL	Specifies the URL of the body file.
CODE_NAME	Specifies the name of the code file.
CODE_TITLE	Specifies the title of the code file.
CODE_URL	Specifies the URL of the code file.
CONTENTS_NAME	Specifies the name of the contents file.
CONTENTS_TITLE	Specifies the title of the contents file.
CONTENTS_URL	Specifies the URL of the contents file.
DATA_NAME	Specifies the name of the data file.
DATA_TITLE	Specifies the title of the data file.

Event Variable	Description
DATA_URL	Specifies the URL of the data file.
ENCODING	Specifies the encoding of the output for converting text data into a numbering system that computers recognize.
FRAME_NAME	Specifies the name of the frame file.
FRAME_TITLE	Specifies the title of the frame file.
FRAME_URL	Specifies the URL of the frame file.
GRAPH_PATH_NAME	Specifies the path of the graph as specified by the ODS PATH statement.
GRAPH_PATH_URL	Specifies the URL of the graph.
NO_BOTTOM	is non-zero if you specified the NO_BOTTOM_MATTER option on the ODS MARKUP statement.
NO_TOP	is non-zero if you specified the NO_TOP_MATTER option on the ODS MARKUP statement.
OPERATOR	Specifies the operator. The value of the OPERATOR event variable is set from an ODS statement or, by default, is the user that is running SAS.
PAGES_NAME	Specifies the name of the pages file.
PAGES_TITLE	Specifies the title of the pages file.
PAGES_URL	Specifies the URL of the pages file.
PATH	Specifies the path as set by an ODS statement.
PATH_NAME	Specifies the path name.
PATH_URL	Specifies the path location.
STYLESHEET_NAME	Specifies the name of the stylesheet file.
STYLESHEET_TITLE	Specifies the title of the stylesheet file.
STYLESHEET_URL	Specifies the URL of the stylesheet file.
TAGSET	Specifies the name of the current tagset.
TAGSET_ALIAS	Specifies the alias of the current tagset as given in the ODS MARKUP statement.
TITLE	Specifies the title from the ODS statement.
TRANTAB	Specifies the translation table name for character conversions.

Table 13.12 Table Variables

Event Variable	Description
CLABEL	Specifies the label for the output object in the contents file, the Results window, and the trace record. Set with the CONTENTS_LABEL= attribute in the table template. For information, see the CONTENTS_LABEL on page 646 attribute.
COLCOUNT	Specifies the number of columns in the current table.
COLEND_EA	Specifies the ending column number.

Event Variable	Description
COLSPAN	Specifies the number of columns that the cell spans.
COLSTART	Specifies the column number where the cell starts.
DATA_ROW	Specifies that the current row is a data row.
IS_STACKED	Specifies that the columns are stacked.
ROW	Specifies the current table row, which includes headers.
ROWSPAN	Specifies the number of rows that the current cell spans.
SECTION	Specifies the header, body, or footer of the table.
WIDTH	Specifies the width. WIDTH is most commonly used for COLSPECS.

Table 13.13 URL Variables

Event Variable	Description
NOBASE	Sets a flag to determine whether to use the value for BASE= option as part of the URL. 0 uses the BASE= option, and 1 does not use BASE= option.
TARGET	Specifies the target that is associated with the URL.
URL	Specifies a fully formed URL.

Concepts: Markup Languages and the TEMPLATE Procedure

Getting Familiar with Tagsets

Listing Tagset Names

SAS provides a set of tagsets. To get a list of the tagsets that SAS supplies and any tagsets that you have created and stored in the SASHELP.TMPLMST template store, submit the following SAS statements:

```
proc template;
   list tagsets;
run;
```

By default, PROC TEMPLATE lists the tagsets in SASHELP.TMPLMST and SASUSER.TEMPLAT. Typically, you have read-only permissions to the SASHELP.TMPLMST item store where the SAS tagset directory is located. The SASUSER.TEMPLAT is the item store where the tagsets that you create or customize are stored by default.

Specifying Tagset Names

To specify a SAS tagset stored in SASHELP.TMPLMST or a tagset that you have created and stored in SASUSER.TEMPLAT or any other item store, use a two-level name: TAGSETS.*tagset-name*. For example, tagsets.chtml or tagsets.mytagset are valid two-level tagset names. By default, SAS knows that the specified tagset is stored in either SASHELP.TMPLMST or SASUSER.TEMPLAT.

To specify a tagset that you have created and stored in an item store other than SASUSER.TEMPLAT, assign the item store to the ODS search path with the ODS PATH statement. For information about the ODS PATH statement, see "ODS PATH Statement" on page 206.

Viewing the Contents of a Tagset

To view the contents of a tagset, use the SAS windowing environment or the TEMPLATE procedure.

- SAS Windowing Environment
  1. From the menu, select **View ▶ Results**.
  2. In the **Results** window, select the **Results** folder. Right-click and select **Templates** to open the Templates window.
  3. Double-click **Tagsets** to view the contents of that item store or directory.
  4. Double-click the tagset that you wish to view. For example, the CHTML tagset is the template store for CHTML output.
- SAS Windowing Command
  1. To view the Templates window, submit the following command in the command bar:

     ```
     odstemplates
     ```

     The Templates window contains the item stores **Sasuser.Templat** and **Sashelp.Tmplmst**.
  2. When you double-click an item store, such as **Sashelp.Tmplmst**, that item store expands to list the directories where ODS templates are stored. The templates that SAS provides are in the item store Sashelp.Tmplmst.
  3. To view the tagsets that SAS provides, double-click the **Tagset** item store.
  4. Right-click the tagset, such as **Rtf**, and select **Open**. The tagset is displayed in the Template Browser window.
- TEMPLATE Procedure
  1. To see the source for a tagset, use PROC TEMPLATE and specify the two-level name of the tagset. For example, to see the source of a SAS tagset that generates CHTML output, submit these SAS statements:

     ```
     proc template;
           source tagsets.chtml;
     ```

     The source for TAGSETS.CHTML consists of the following:
     - a DEFINE TAGSET statement that names the tagset
     - events that define what is written to the output file
     - tagset attributes, such as output type and the character to use for line breaks

Understanding Events

A tagset controls output generation through a series of events and variables. An event defines what is written to the output file. Here are some key points about events:

- Events have unique names. SAS procedures that generate ODS output use a standard set of events, which you can customize by redefining them in the customized tagset. In addition, you can define custom events.
- The DEFINE EVENT statement assigns a name to an event.
- An event can include start sections, finish sections, or both. These sections specify different actions. If the event does not include either a start or finish section, then the event is stateless: no matter how the event is called, all of the actions in the event are executed. If an event has a finish section, then a start section is assumed if there are statements above the finish section.
- An event can execute another event using the TRIGGER statement. From the start section of an event, any event triggered also runs its start section. From the finish section, the triggered event runs its finish section. If a triggered event does not have start or finish sections, then the event runs the statements that it does have. A trigger can also explicitly ask for an event's specific section. See Example 4 on page 857.
- Events can perform actions based on conditions.
- For the most part, an event consists of PUT statements, text, and event variables.

For example, here is a simple event for an HTML table output:

```
define event table;❶
start:❷
      put '<table>' nl;
finish:
      put '</table>' nl;
end;
```

In the event:

❶ The DEFINE EVENT statement begins the event and assigns it the name TABLE.

❷ The START section defines the beginning portion of the event, and the FINISH section defines the ending portion of the event. An event for a table needs START and FINISH sections because ODS needs to know how to define the beginning and the ending. ODS also expects other events to define how to format the table's rows and columns. The PUT statements specify to write the tags **<table>** and **</table>** to the output file, and to add a new line after each tag.

The following event does not include a start and finish section. The PUT statements specify to write the tags **<TD>** and **</TD>** to the output file. In addition, the event variable VALUE is used so that the data value from the SAS procedure or data set is written to the output file. The data value is enclosed with the **<TD>** and **</TD>** tags.

```
define event data;
      put '<TD>';
      put VALUE;
      put '</TD>';
end;
```

Understanding Variables

A variable is a programming structure that is used to hold data. A variable holds the data that is assigned to it until you assign a new value or end the program. Each variable has a unique name and holds information that is either internal information to handle the requested output (metadata that is used by ODS or the XML LIBNAME engine) or is information that is directly related to the output itself. For example, the variable COLCOUNT holds the value for the number of columns in the output, and the variable DATE holds the date.

Variables that are used by tagsets are divided into two groups: internally generated and user-created.

There are three logical divisions of internally generated variables:

event variable a variable that includes text, formatting, and data values. These variables can originate in many places, such as the table template, the procedure, the title, or byline processing.

style variable a variable that specifies a value for one aspect of the presentation. Style variables are specified by the ODS style attributes that are currently in use. The style variables are only differentiated from other event variables in that you know exactly where they originate. For more information on style attributes, see Chapter 11, "TEMPLATE Procedure: Creating a Style Template (Definition)," on page 487.

dynamic variable a variable that is dynamically created within SAS. Because these variables are dynamically created, their names, or how they are used, is unknown. These variables are dynamic because they are not defined by ODS, but by applications such as SAS/GRAPH and the XML LIBNAME engine. Dynamic variables are designated by a preceding @ symbol. Dynamic variables are listed with the DYNAMIC statement. For more information about SAS/GRAPH, see *SAS/GRAPH: Reference*.

There are five types of user-created variables:

dictionary variable an array that contains a list of numbers or text strings that are identified by a key. A dictionary variable has, as part of its name, a preceding '$' symbol and a subscript that contains a text string or a variable that has a character value. The text string or variable within the subscript is called a key. Keys are case preserving and case sensitive. After dictionary variables are created, they are globally available in all events and persist until you unset them with the UNSET statement.

For example, the following dictionary variable is identifying the entry in the $MyDictionary variable that contains the text 'dog': $MyDictionary['dog']. In this example, the key is 'dog'. Dictionary variables are accessed sequentially by using the ITERATE and NEXT statements.

list variable an array that contains a list of numbers or text strings that are indexed. A list variable has, as part of its name, a preceding '$' symbol and a subscript that is empty or contains a number or numeric variable. The number within the subscript is called an index. After they are created, list variables are globally available in all events and persist until you unset them with the UNSET statement.

List entries are accessed by positive or negative indexes. Positive indexes start at the beginning of a list. Negative indexes start at the end of a list. For example, the list variable $Mylist[2] identifies the second entry in the list variable $Mylist. In this case, the index is 2. The list variable $Mylist[-2] identifies the second entry from the end of the list variable $Mylist. In this case, the index is [-2]. List variables are accessed sequentially by using the ITERATE and NEXT statements.

macro variable a variable that is part of the SAS macro programming language. Macro variables must be specified with the MVAR or NMVAR statements. After they are declared, macro variables can be used anywhere within an event. See the “MVAR Statement” on page 620 and “NMVAR Statement” on page 621 for more information.

memory variables areas of memory that contain numeric data, character data, or lists of numeric or character data. A memory variable is classified as a dictionary variable if it is created with a subscript that contains a key. A memory variable is classified as a list variable if it is created with a subscript that is empty or contains an index. If you omit a key or an index, then the memory variable is a numeric or character scalar variable, depending on the variable's value.

scalar variable an area of memory that contains numeric or character data. Scalar variables must be preceded by the '$' symbol. After scalar variables are created, they are globally available in all events and persist until you unset them with the UNSET statement.

stream variable a temporary item store that contains output. All output from PUT statements is directed to the open stream variable until it is closed. Stream variables must be preceded by the '$$' symbol except when used with the OPEN or PUTSTREAM statements. Stream variables are created with the SET, EVAL, or OPEN statements, within the DEFINE EVENT statement. Stream variables are different from other variables in that they can hold very large amounts of data. They can hold very large amounts of data because as they increase in size they are written to disk as needed.

Displaying Event Variables and Their Values

Because variables represent data, their values might or might not be present, depending on the SAS procedure and the job. For example, some variables have values only if they are specified with procedure options or style options. Other variables have values because the internal information, such as how many columns are in the output, is needed. For example, TAGSETS.CHTML contains the event COLSPECS, which uses the event variable COLCOUNT so that ODS knows how many columns are in the output:

```
define event colspecs;
      put '<p>' nl '<table';
      putq ' columns=' COLCOUNT;
      put ' cellpadding=2 border=1>' nl;
end;
```

To determine which variables have values and what the values are, use the EVENT_MAP statement to submit the SAS program. For more information, see “ Defining a Tagset Using the EVENT_MAP Tagset” on page 843. For a list of event variables and their descriptions, see “Event Variables” on page 833.

Creating Custom Tagsets

Methods for Creating Custom Tagsets

To create a tagset, use the TEMPLATE procedure to define the tagset. In general, three methods are available to create a custom tagset:

- Define a tagset through inheritance.
- Copy an existing tagset, and then modify it.
- Define a custom tagset.

Inheriting Events in a Tagset

Tagsets can inherit events from each other. For example, the SAS tagset TAGSETS.WMLOLIST inherits most of its events from TAGSETS.WML, and TAGSETS.IMODE gets most of its events from TAGSETS.CHTML. Inheriting events from an existing tagset is the easiest way to define a new tagset.

To inherit events, a tagset uses the PARENT= attribute in the DEFINE TAGSET statement to specify the name of a tagset from which to inherit. When a parent is specified for a tagset, all of the tagset options, attributes, and statements that are specified in the parent's template are used in the new template, unless the new template overrides them. That is, in the new tagset, an event can override the operation of the same-named event that is defined in the parent tagset. For example, if the parent tagset defines an event named TABLE, then you can change the operation in the new tagset by redefining the event named TABLE.

For an example of inheriting events in a tagset, see Example 1 on page 846.

Defining a Tagset Using the EVENT_MAP Tagset

SAS procedures that generate ODS output use a standard set of events and variables. To generate customized output, create a customized tagset with customized events. However, in order to customize the events, you need to know the names of the events that ODS uses.

A good way to start defining the customized tagset is to use the EVENT_MAP tagset that SAS supplies. This enables you to determine which events are triggered and which variables are used by an event to send output from a SAS process to an output file. When you run a SAS process with TAGSETS.EVENT_MAP, ODS writes XML markup to an output file that shows all event names and variable names as tags. In the output, tag names are the event names. Tag attributes are the variables that have values for those events.

For example, the following statements run ODS MARKUP with TYPE=EVENT_MAP to see which events and variables ODS uses for various parts of the PROC PRINT output:

```
ods markup type=event_map file='custom-tagset-filename.xml';

   proc print data=sashelp.class;
      where Height gt 60;
   run;

   ods markup close;
```

Here is the listing output and resulting XML file:

Output 13.1 Listing Output

The SAS System 1

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
12	Judy	F	14	64.3	90.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
19	William	M	15	66.5	112.0

Output 13.2 XML File

```
<?xml version='1.0' encoding='windows-1252'?>

<doc operator='user' sasversion='9.1' saslongversion='9.01.01B0D06102003'
      date='2003-06-11' time='15:55:02' encoding='windows-1252' event_name='doc'
      trigger_name='attr_out' class='Body' index='IDX' just='c'>
  <doc_head event_name='doc_head' trigger_name='attr_out' class='Body'
      index='IDX' just='c'>
    <doc_meta event_name='doc_meta' trigger_name='attr_out' class='Body'
      index='IDX' just='c'/>
    <auth_oper event_name='auth_oper' trigger_name='attr_out' class='Body'
      index='IDX' just='c'/>
    <doc_title event_name='doc_title' trigger_name='attr_out' class='Body'
      index='IDX' just='c'/>
    <stylesheet_link event_name='stylesheet_link' trigger_name='attr_out'
      index='IDX' just='c'/>
    <javascript event_name='javascript' trigger_name='attr_out' class='Body'
      index='IDX' just='c'>
      <startup_function event_name='startup_function' trigger_name='attr_out'
         class='StartUpFunction' index='IDX' just='c'>
      </startup_function>
      <shutdown_function event_name='shutdown_function' trigger_name='attr_out'
         class='ShutDownFunction' index='IDX' just='c'>
      </shutdown_function>
    </javascript>
  </doc_head>
  <doc_body event_name='doc_body' trigger_name='attr_out' class='Body'
      index='IDX' just='c'>
    <proc event_name='proc' trigger_name='attr_out' name='Print'
      index='IDX' just='c'>
      <anchor event_name='anchor' trigger_name='attr_out' class='Body' name='IDX'
         index='IDX' just='c'/>
      <page_setup event_name='page_setup' trigger_name='attr_out' class='Body'
         index='IDX' just='c'>
        <system_title_setup_group event_name='system_title_setup_group' trigger_name='attr_out'
          class='Body' colcount='1' index='IDX' just='c'>
          <title_setup_container event_name='title_setup_container' trigger_name='attr_out'
            class='SysTitleAndFooterContainer' colcount='1' index='IDX' just='c'>
            <title_setup_container_specs event_name='title_setup_container_specs' trigger_name='attr_out'
               class='SysTitleAndFooterContainer' colcount='1' index='IDX' just='c'>
              <title_setup_container_spec event_name='title_setup_container_spec' trigger_name='attr_out'
                  colcount='1' type='string' index='IDX' just='c' width='100%'/>
            </title_setup_container_specs>
            <title_setup_container_row event_name='title_setup_container_row' trigger_name='attr_out'
               colcount='1' index='IDX' just='c'>
              <system_title_setup event_name='system_title_setup' trigger_name='attr_out'
                  class='SystemTitle' value='The SAS System' colcount='1' index='IDX' just='c'>
              </system_title_setup>
            </title_setup_container_row>
          </title_setup_container>
        </system_title_setup_group>
      </page_setup>

...more xml tagged output...

               </table_body>
             </table>
           </output>
         </leaf>
      </proc_branch>
    </proc>
  </doc_body>
</doc>
```

In the XML output that is generated by EVENT_MAP, PROC PRINT uses events named DOC_HEAD, PROC, TABLE, and so on. The TABLE event uses data from event variables such as STATE, CLASS, and TYPE. After you know the events and variables

that generate the output, define the tagset and customize your events. For example, you could redefine the TABLE event to produce customized output.

To define a tagset with which to customize your output, start by specifying TAGSETS.EVENT_MAP as the parent tagset. As you redefine events to customize output, these events replace the default events that are defined in the EVENT_MAP tagset. In addition, you can remove the operation of a default event by redefining it as an empty event in the tagset. When you are satisfied with the customized output, remove the EVENT_MAP inheritance and the empty events. Then the output will reflect only the events you defined.

Note: When you first run a SAS process and specify TYPE=EVENT_MAP, you can also generate a stylesheet along with the body file. The stylesheet shows which style attributes you are using. △

Alternatives to EVENT_MAP

To create other types of output, you can use one of the following tagsets as alternatives:

- TEXT_MAP generates output that is similiar to a listing output.
- TPL_STYLE_LIST generates HTML and TPL_STYLE_MAP generates XML. However, these tagsets list only a subset of the possible attributes.
- STYLE_POPUP generates HTML like HTMLCSS. However Internet Explorer, STYLE_POPUP displays a window that shows the resolved ODS style definition for any item that you click.
- STYLE_DISPLAY is like STYLE_POPUP, but it generates a simple page of output for you to click.
- NAMEDHTML generates HTML output like STYLE_POPUP, but all of the objects are labeled the same as with ODS TRACE.

Defining a Tagset Using SAS DATA Step Functions

A SAS DATA step function performs a computation or system manipulation on arguments and returns a value. In Base SAS software, you can use SAS functions in DATA step programming statements, WHERE expressions, macro language statements, the REPORT procedure, Structured Query Language (SQL), and in statements that are used when creating custom tagsets. Functions can be used on any statement within the tagset language. For information on DATA step functions and statements, see *SAS Language Reference: Dictionary* and *SAS Language Reference: Concepts*.

Examples: Creating and Modifying Markup Languages Using the TEMPLATE Procedure

Example 1: Creating a Tagset through Inheritance

PROC TEMPLATE features:

DEFINE TAGSET statement:

DEFINE EVENT statement:

PUT statement

PARENT= attribute

Other ODS features:

ODS PATH statement

ODS MARKUP statement

Program Description

This example defines a new tagset called TAGSETS.MYTAGS that creates customized HTML output. The new tagset is created through inheritance. Most of the required formatting is available in the tagset TAGSETS.CHTML, which SAS supplies.

Program

Define a new tagset. The DEFINE TAGSET statement creates a new tagset called **tagsets.mytags**. The PARENT= attribute is used so that the new tagset **tagsets.mytags** inherits events from TAGSETS.CHTML. Note that the ODS PATH statement is specified at the beginning to establish the search path.

```
ods path sasuser.templat (update)
   sashelp.tmplmst (read);

proc template;
   define tagset tagsets.mytags /store=sasuser.templat;
      parent=tagsets.chtml;
```

Define three events. The DEFINE EVENT statements create three events called **colspecs, table,** and **system_title**. The **colspecs** event specifies text. The **table** event specifies tags to include in the template. The **system_title** event deletes titles.

```
define event colspecs;
   put 'These are my new colspecs' nl;
end;

define event table;
   put '<p>' nl '<table>';
finish:
   put '</table>';
end;

define event system_title;
end;
```

End the tagset. This END statement ends the tagset. The RUN statement executes the PROC TEMPLATE step.

```
end;
run;
```

Specify the user-defined tagset. The following code specifies the user-defined tagset TAGSETS.MYTAGS as the tagset for the output.

```
ods tagsets.mytags body='custom-tagset-filename.html';
```

Print the data set. PROC PRINT creates the report. ODS writes the report to the body file.

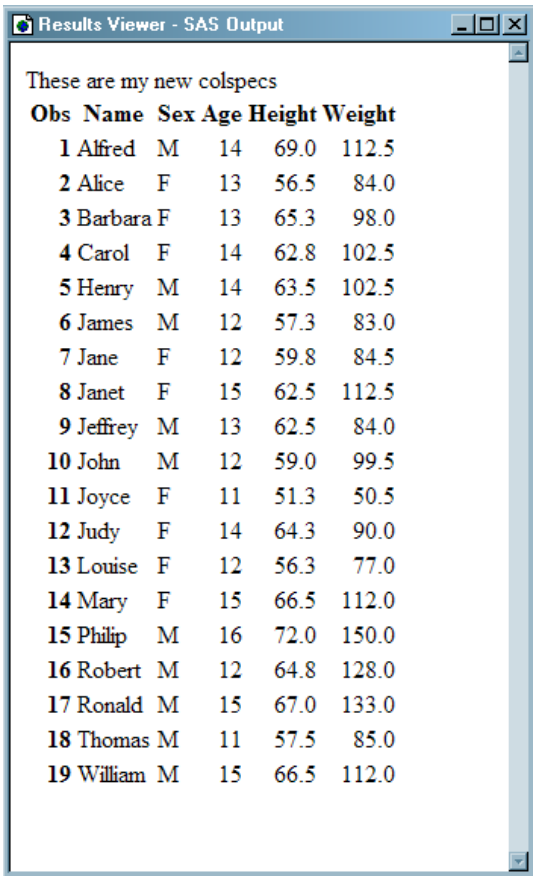
```
proc print data=sashelp.class;
   run;
```

Stop the creation of the tagset. The ODS TAGSET. MYTAGS CLOSE statement closes the MARKUP destination and all the files that are associated with it. Close the destination so that you can view the output with a browser.

```
ods tagsets.mytags close;
```

Display 13.1 Generated Output: MYTAGS.CHTML (Viewed with Microsoft Internet Explorer)

To see the customized CHTML tags, view the source from the Web browser:
From the browser's tool bar, select **View ► Source**.



Use the tagset TAGSETS.CHTML, which SAS provides. Compare the output from TAGSETS.MYTAGS to that from TAGSETS.CHTML, which SAS supplies. Use the following ODS code to specify the SAS tagset. You can specify any tagset by using TYPE= in an ODS MARKUP statement.

```
ods markup type=tagsets.chtml body='default-tagset-filename.html';

   proc print data=sashelp.class;
   run;

   ods markup close;
```

Display 13.2 A Display That Uses the Default CHTML Tagset (Viewed with Microsoft Internet Explorer)

To see the default CHTML tags, view the source from the Web browser:
Ffrom the browser's tool bar, select **View ► Source**.

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Example 2: Creating a Tagset by Copying a Tagset's Source

PROC TEMPLATE features:
SOURCE statement
DEFINE TAGSET statement

DEFINE EVENT statement

Program Description

This example copies the source for a tagset that SAS supplies, modifies the template, and then builds a new tagset for custom output. To create a new tagset, use the SOURCE statement in PROC TEMPLATE to copy a tagset's source. Then you can customize the template as necessary.

Program

Copy the SAS tagset to an external file. The following statements copy the tagset source from the SAS tagset TAGSETS.CSV to the SAS log.

```
proc template;
      source tagsets.csv;
   run;
```

Output 13.3 CSV Tagset Source

This is the default CSV tagset that SAS supplies.

```
define tagset Tagsets.Csv;
   notes 'This is the CSV template';
   define event put_value;
      put VALUE;
   end;
   define event put_value_cr;
      put VALUE NL;
   end;
   define event table;
      finish:
         put NL;
   end;
   define event row;
      finish:
         put NL;
   end;
   define event header;
      start:
         put ',' /if ^cmp( COLSTART, '1');
         put '''';
         put VALUE;
      finish:
         put '''';
   end;
   define event data;
      start:
         put ',' /if ^cmp( COLSTART, '1');
         put '''';
         put VALUE;
      finish:
         put '''';
   end;
   define event colspanfill;
      put ',';
   end;
   define event rowspanfill;
      put ',' /if ^exists( VALUE);
   end;
   define event breakline;
      put NL;
   end;
   define event splitline;
      put NL;
   end;
   registered_tm = '(r)';
   trademark = '(tm)';
   copyright = '(c)';
   output_type = 'csv';
   stacked_columns = OFF;
end;
```

Create the customized tagset. Submit the following PROC TEMPLATE code to create the customized tagset **`Tagsets.mycsv`**. The DEFINE EVENT TABLE statement uses the PUT NL statements to add two blank lines to the output file. One blank line is placed before the table and the other is placed after the table.

```
define tagset Tagsets.mycsv;
   notes 'This is the My CSV template';
   define event table;
```

```
      start:
         put nl;
      finish:
         put nl;
   end;
   define event put_value;
      put VALUE;
   end;
   define event put_value_cr;
      put VALUE NL;
   end;
   define event row;
      finish:
         put NL;
   end;
   define event header;
      start:
         put ',' /if ^cmp( COLSTART, '1');
         put '''';
         put VALUE;
      finish:
         put '''';
   end;
   define event data;
      start:
         put ',' /if ^cmp( COLSTART, '1');
         put '''';
         put VALUE;
      finish:
         put '''';
   end;
   define event colspanfill;
      put ',';
   end;
   define event rowspanfill;
      put ',' /if ^exists( VALUE);
   end;
   define event breakline;
      put NL;
   end;
   define event splitline;
      put NL;
   end;
   registered_tm = '(r)';
   trademark = '(tm)';
   copyright = '(c)';
   output_type = 'csv';
   stacked_columns = OFF;
end;
```

Output 13.4 Customized CSV Tagsets.mycsv Template Source

To view the customized CSV **`Tagsets.mycsv`**, submit the following code:

```
proc template;
   source tagsets.mycsv;
run;
```

```
proc template;
   define tagset Tagsets.Mycsv / store = SASUSER.TEMPLAT;
      notes 'This is the My CSV template';
      define event table;
         start:
            put NL;
         finish:
            put NL;
      end;
      define event put_value;
         put VALUE;
      end;
      define event put_value_cr;
         put VALUE NL;
      end;
      define event row;
         finish:
            put NL;
      end;
      define event header;
         start:
            put ',' /if ^cmp( COLSTART, '1');
            put '''';
            put VALUE;
         finish:
            put '''';
      end;
      define event data;
         start:
            put ',' /if ^cmp( COLSTART, '1');
            put '''';
            put VALUE;
         finish:
            put '''';
      end;
      define event colspanfill;
         put ',';
      end;
      define event rowspanfill;
         put ',' /if ^exists( VALUE);
      end;
      define event breakline;
         put NL;
      end;
      define event splitline;
         put NL;
      end;
      output_type = 'csv';
      copyright = '(c)';
      trademark = '(tm)';
      registered_tm = '(r)';
      stacked_columns = OFF;
   end;
run;
```

Example 3: Creating a New Tagset

PROC TEMPLATE features:

DEFINE TAGSET statement:
- NOTES statement
- DEFINE EVENT statement:
  - NDENT statement
  - PUT statement
  - TRIGGER statement
  - XDENT statement

Tagset Attributes:

DEFAULT_EVENT attribute
INDENT= attribute
OUTPUT_TYPE attribute
MAP= attribute
MAPSUB= attribute
NOBREAKSPACE= attribute
SPLIT= attribute
STACKED_COLUMNS= attribute

Program Description

This example shows a new tagset that does not inherit events from another tagset. This is a customized tagset for specific PROC FREQ output.

Program

Create the new tagset Tagsets.newloc. The DEFINE TAGSET statement creates a new tagset **`Tagsets.newloc`** and specifies where you want to store the tagset.

```
proc template;
   define tagset Tagsets.newloc / store = SASUSER.TEMPLAT;
      notes 'This is the Location Report Template';
```

Define seven events. The seven DEFINE statements create the events named **`basic, doc, system_title, header, data, country,`** and **`frequency`**.

```
   define event basic;
   end;

   define event doc;
   start:
      put '' nl nl;
      put '' nl;
      put '' nl;
      put '' nl;
      ndent;
   finish:
      xdent;
```

```
      put nl;
      put '';
   end;

   define event system_title;
      put '';
      put VALUE;
      put '';
      put nl nl;
   end;
define event header;
   start:
   trigger country /if cmp(LABEL, 'EmpCountry');
   end;

   define event data;
   start:
   trigger frequency /if cmp(name, 'Frequency');
   end;

   define event country;
      put '' nl ;
      ndent ;
      put '' ;
      put VALUE ;
      put '' nl  ;
   end;

   define event frequency;
      put '' ;
      put VALUE ;
      put '' nl  ;
      xdent ;
      put '' nl ;
   end;

   output_type = 'xml';
   default_event = 'basic';
   indent = 2;
   split = '';
   nobreakspace = ' ';
   mapsub = '/</>/&/';
   map = '<>&';
   stacked_columns=off;
   end;
run;
```

Output 13.5 New Tagsets.newloc Template Source

```
proc template;
   define tagset Tagsets.newloc / store = SASUSER.TEMPLAT;
      notes 'This is the Location Report Template';
      define event basic;
      end;
      define event doc;
         start:
            put '' NL NL;
            put '' NL;
            put '' NL;
            put '' NL;
            ndent;
         finish:
            xdent;
            put NL;
            put '';
      end;
      define event system_title;
         put '';
         put VALUE;
         put '';
         put NL NL;
      end;
      define event header;
         start:
            trigger country /if cmp( LABEL, 'EmpCountry');
      end;
      define event data;
         start:
            trigger frequency /if cmp( name, 'Frequency');
      end;
      define event country;
         put '' NL;
         ndent;
         put '';
         put VALUE;
         put '' NL;
      end;
      define event frequency;
         put '';
         put VALUE;
         put '' NL;
         xdent;
         put '' NL;
      end;
      map = %nrstr('<>&');
      mapsub = %nrstr('//&/');
      nobreakspace = ' ';
      split = '';
      indent = 2;
      default_event = 'basic';
      output_type = 'xml';
      stacked_columns = OFF;
   end;
run;
```

Example 4: Executing Events Using the TRIGGER= Statement

PROC TEMPLATE features:

DEFINE TAGSET statement:

DEFINE EVENT statement:

PUT statement

TRIGGER statement

Other ODS features:

ODS *directory.tagset-name* statement

Program Description

This example illustrates how to execute events.

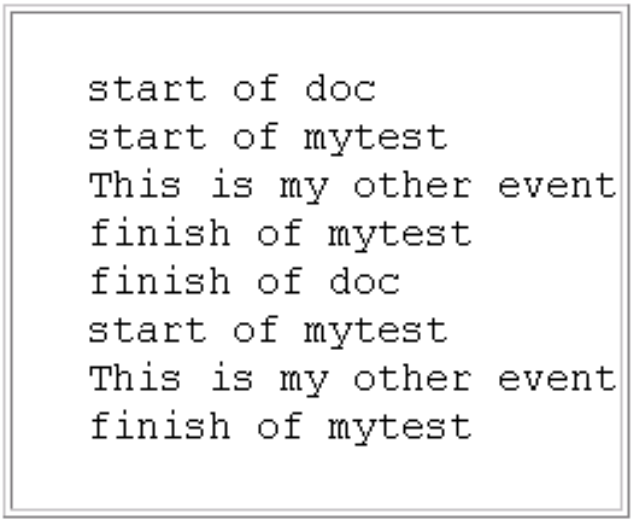
Program

Execute different events. The TRIGGER statement executes another event. For example, the start section of DOC triggers the start section of MYTEST and OTHEREVENT. MYTEST has a start section, so output is generated. OTHEREVENT is stateless (no start or finish sections), but output is generated.

```
proc template;
   define tagset tagsets.mytagset;
      define event doc;
      start:
         put 'start of doc' nl;
         trigger mytest;
         trigger otherevent;
      finish:
         trigger mytest;
         put 'finish of doc' nl;
         trigger mytest start;
         trigger otherevent;
         trigger mytest finish;
      end;

      define event mytest;
      start:
         put 'start of mytest' nl;
      finish:
         put 'finish of mytest' nl;
      end;

      define event otherevent;
         put 'This is my other event' nl;
      end;
   end;
run;

ods tagsets.mytagset file='custom-tagset-filename.txt';
ods tagsets.mytagset close;
```

Display 13.3 Output Created from Events and Tagsets.mytagset Template

To view the output **tagsets.mytagset**, open the file in a text editor.



```
start of doc
start of mytest
This is my other event
finish of mytest
finish of doc
start of mytest
This is my other event
finish of mytest
```

Example 5: Indenting Output

PROC TEMPLATE features:

DEFINE TAGSET statement:

DEFINE EVENT statement:

PUT statement
NDENT statement
TRIGGER statement
XDENT statement

TAGSET attributes:

INDENT= attribute

Other ODS features:

ODS *directory.tagset-name* statement

Program Description

This example illustrates how to indent the output using a tagset.

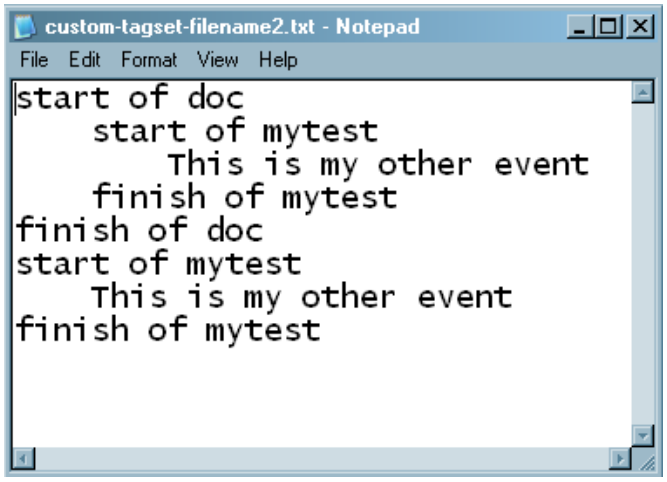
Note: When you view a file with an extension of .xml in an XML-compliant browser, the browser ignores any indention in the file in favor of its own indention algorithm. △

Program

Set the beginning indention level and then proceed to increment the indention levels. The INDENT= tagset attribute determines how much the NDENT and XDENT event statements indent output.

```
proc template;
   define tagset tagsets.mytagset2;
   indent = 4;

      define event doc;
      start:
         put 'start of doc' nl;
         ndent;
         trigger mytest;
         trigger otherevent;
      finish:
         trigger mytest;
         xdent;
         put 'finish of doc' nl;
         trigger mytest start;
         trigger otherevent;
         trigger mytest finish;
      end;

      define event mytest;
      start:
         put 'start of mytest' nl;
         ndent;
      finish:
         xdent;
         put 'finish of mytest' nl;
      end;

      define event otherevent;
         put 'This is my other event' nl;
      end;
   end;
run;
ods tagsets.mytagset2 file='custom-tagset-filename2.txt';
ods tagsets.mytagset2 close;
```

Display 13.4 Output Created from Events and Using Tagsets.mytagset2 Template Source



Example 6: Using Different Styles for Events

PROC TEMPLATE features:

DEFINE EVENT statement:

PUT statement
TRIGGER statement

Event attribute:

STYLE= attribute

Program Description

This example uses different styles for events.

Program

Specify the events. The following events are from the SAS tagset TAGSETS.HTMLCSS, and they show how ODS creates notes. By defining the Gnote event and setting the proper style in the right place, ODS creates a two-cell table that has a banner using the appropriate banner style and a content cell that has the appropriate content style.

```
define event Gnote;
           start:
               put '<div>';
               trigger align;
               put '>';
               put '<table>';
               put '<tr>' nl;
           finish:
               put '</tr>' nl;
               put '</table>' nl;
               put '</div>';
       end;
```

```
define event GBanner;
    put '' nl;
    trigger pre_post;
    put '' nl;
end;

define event GNContent;
    put '';
    trigger pre_post start;
    put VALUE;
    trigger pre_post finish;
    put '';
end;

define event noteBanner;
    style=NoteBanner;
    trigger GBanner;
end;

define event NoteContent;
    style=NoteContent;
    trigger GNContent;
end;

define event note;
    trigger Gnote start;
    trigger noteBanner;
    trigger noteContent;
    trigger Gnote finish;
end;

define event WarnBanner;
    style=WarnBanner;
    trigger GBanner;
end;

define event WarnContent;
    style=WarnContent;
    trigger GNContent;
end;

define event Warning;
    trigger Gnote start;
    trigger WarnBanner;
    trigger WarnContent;
    trigger Gnote finish;
end;
```

Example 7: Modifying an Event to Include Other Stylesheets

PROC TEMPLATE features:

DEFINE EVENT statement:

PUTQ statement

Program Description

The following program provides some example code that you can use to link a previously created stylesheet to an event that you define.

Program

Define an event that links to a stylesheet. This code defines an event that creates a link to a previously created stylesheet instead of the stylesheet that SAS generated.

```
define event stylesheet_link;
putq '<link rel= 'STYLESHEET' type='text/css'
href=' URL '>' nl / if exists(url);
putq '<link rel= 'STYLESHEET' type='text/css'
href='http://your/stylesheet/url/goes/here'>' nl;
putq '<link rel= 'STYLESHEET' type='text/css'
href='http://your/stylesheet/url/goes/here'>' nl;
end;
```

Example 8: Using the STACKED_COLUMNS Attribute in a Tagset

PROC TEMPLATE features:

DEFINE TABLE statement:

NOTES statement
COLUMN statement
DEFINE statement (for columns)

DEFINE TAGSET statement:

Tagset attribute:

PARENT= attribute
STACKED_COLUMNS= attribute

Other ODS features:

ODS *directory.tagset-name* statement

ODS PHTML statement

ODS _ALL_ CLOSE statement

Program Description

This example shows the difference between stacking data one column on top of another and placing data side by side. (For more information on stacked columns, see the "DEFINE TABLE Statement" on page 640.)

Program

Create a table template. The DEFINE TABLE statement creates the table template **`Base.Standard`**.

```
proc template;
   define table Base.Standard;
      notes 'Table template for PROC Standard.';
      column name (mean std) n label;
      define name;  header='Name' varname='Name' style=RowHeader; end;
      define mean;  header='Mean/Std Dev' varname='Mean' format=D12.;
end;
      define std;   header='/Standard/Deviation'
         varname='stdDev' format=D12.; end;
      define n;     header='N' format=best.; end;
      define label; header='Label' varname='Label'; end;
      byline wrap required_space=3;
   end;
run;
 proc template;
   define tagset tagsets.myhtml;
        parent=tagsets.phtml;
        stacked_columns=no;
   end;
run;
```

Customize the tagset by stacking the values side by side. This customized tagset has STACKED_COLUMNS= NO. Note that the SAS tagset, TAGSETS.PHTML, has STACKED_COLUMNS=YES.

```
proc template;
   define tagset tagsets.myhtml;
        parent=tagsets.phtml;
        stacked_columns=no;
   end;
run;
```

Create HTML output and specify the location for storing the HTML output. The ODS TAGSETS.MYHTML statement opens the markup language destination and creates the HTML output. The output objects are sent to the external file **`not_stacked.html`** in the current directory. The PROC STANDARD statement generates the statistics for the **`sashelp.class`** data set. The PRINT option prints the report.

```
ods tagsets.myhtml file='not_stacked.html';
proc standard print data=sashelp.class;
run;
```

Stop the creation of the HTML output. The ODS _ALL_ CLOSE statement closes all open destinations and all files associated with them. For HTML output, close the HTML destination so that you can view the output with a browser.

```
ods _all_ close;
```

Display 13.5 Output with Values Side by Side

The SAS System

The STANDARD Procedure

Name	Mean/Std Dev	Standard Deviation	N
Age	13.315789	1.492672	19
Height	62.336842	5.127075	19
Weight	100.026316	22.773933	19

Create the same file but with stacked values. The STACKED_COLUMNS=YES statement shows the same values stacked in the SAS tagset PHTML.

```
ods phtml file='stacked.html';
proc standard print data=sashelp.class;
run;
ods _all_ close;
```

Program

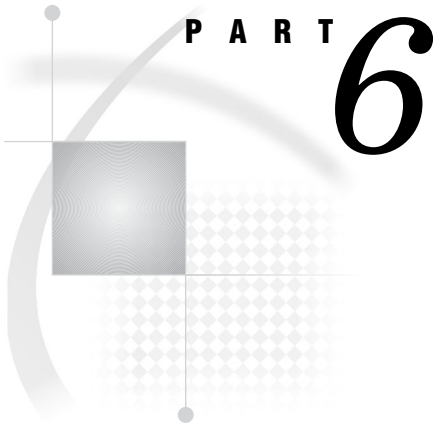
Program Output

Display 13.6 Output with Values Stacked One on Top of Another

The SAS System

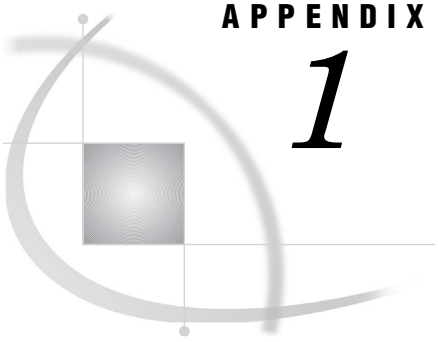
The STANDARD Procedure

Name	Mean/Std Dev	N
Age	13.315789 1.492672	19
Height	62.336842 5.127075	19
Weight	100.026316 22.773933	19



PART 6

Appendices



APPENDIX

1

Example Programs

Creating the $CNTRY Format

```
proc format;
   value $cntry 'BRZ'='Brazil'
                'CHN'='China'
                'IND'='India'
                'INS'='Indonesia'
                'USA'='United States';
run;
```

Creating the Charity Data Set

```
data Charity;
input School $ 1-7 Year 9-12 Name $ 14-20 moneyRaised 22-26
   hoursVolunteered 28-29;
   format moneyRaised dollar8.2;
```

```
   format hoursVolunteered f3.0;
   format Year yrFmt.;
   format School schFmt.;
   label School = "Schools";
   label Year = "Years";
   retain yearmin yearmax;
   yearmin=min(yearmin,year);
   yearmax=max(yearmax,year);
   call symput('first_year',put(yearmin,4.));
   call symput('last_year', put(yearmax,4.));
datalines;
Monroe  1992 Allison 31.65 19
Monroe  1992 Barry   23.76 16
Monroe  1992 Candace 21.11  5
Monroe  1992 Danny    6.89 23
Monroe  1992 Edward  53.76 31
Monroe  1992 Fiona   48.55 13
Monroe  1992 Gert    24.00 16
Monroe  1992 Harold  27.55 17
Monroe  1992 Ima      5.98  9
Monroe  1992 Jack    20.00 23
Monroe  1992 Katie   22.11  2
Monroe  1992 Lisa    18.34 17
Monroe  1992 Tonya   55.16 40
Monroe  1992 Max     26.77 34
Monroe  1992 Ned     28.43 22
Monroe  1992 Opal    32.66 14
Monroe  1993 Patsy   18.33 18
Monroe  1993 Quentin 16.89 15
Monroe  1993 Randall 12.98 17
Monroe  1993 Sam     15.88  5
Monroe  1993 Tyra    21.88 23
Monroe  1993 Myrtle  47.33 26
Monroe  1993 Frank   41.11 22
Monroe  1993 Cameron 65.44 14
Monroe  1993 Vern    17.89 11
Monroe  1993 Wendell 23.00 10
Monroe  1993 Bob     26.88  6
Monroe  1993 Leah    28.99 23
Monroe  1994 Becky   30.33 26
Monroe  1994 Sally   35.75 27
Monroe  1994 Edgar   27.11 12
Monroe  1994 Dawson  17.24 16
Monroe  1994 Lou      5.12 16
Monroe  1994 Damien  18.74 17
Monroe  1994 Mona    27.43  7
Monroe  1994 Della   56.78 15
Monroe  1994 Monique 29.88 19
Monroe  1994 Carl    31.12 25
Monroe  1994 Reba    35.16 22
Monroe  1994 Dax     27.65 23
Monroe  1994 Gary    23.11 15
Monroe  1994 Suzie   26.65 11
Monroe  1994 Benito  47.44 18
```

```
Monroe  1994 Thomas  21.99 23
Monroe  1994 Annie   24.99 27
Monroe  1994 Paul    27.98 22
Monroe  1994 Alex    24.00 16
Monroe  1994 Lauren  15.00 17
Monroe  1994 Julia   12.98 15
Monroe  1994 Keith   11.89 19
Monroe  1994 Jackie  26.88 22
Monroe  1994 Pablo   13.98 28
Monroe  1994 L.T.    56.87 33
Monroe  1994 Willard 78.65 24
Monroe  1994 Kathy   32.88 11
Monroe  1994 Abby    35.88 10
Kennedy 1992 Arturo  34.98 14
Kennedy 1992 Grace   27.55 25
Kennedy 1992 Winston 23.88 22
Kennedy 1992 Vince   12.88 21
Kennedy 1992 Claude  15.62  5
Kennedy 1992 Mary    28.99 34
Kennedy 1992 Abner   25.89 22
Kennedy 1992 Jay     35.89 35
Kennedy 1992 Alicia  28.77 26
Kennedy 1992 Freddy  29.00 27
Kennedy 1992 Eloise  31.67 25
Kennedy 1992 Jenny   43.89 22
Kennedy 1992 Thelma  52.63 21
Kennedy 1992 Tina    19.67 21
Kennedy 1992 Eric    24.89 12
Kennedy 1993 Bubba   37.88 12
Kennedy 1993 G.L.    25.89 21
Kennedy 1993 Bert    28.89 21
Kennedy 1993 Clay    26.44 21
Kennedy 1993 Leeann  27.17 17
Kennedy 1993 Georgia 38.90 11
Kennedy 1993 Bill    42.23 25
Kennedy 1993 Holly   18.67 27
Kennedy 1993 Benny   19.09 25
Kennedy 1993 Cammie  28.77 28
Kennedy 1993 Amy     27.08 31
Kennedy 1993 Doris   22.22 24
Kennedy 1993 Robbie  19.80 24
Kennedy 1993 Ted     27.07 25
Kennedy 1993 Sarah   24.44 12
Kennedy 1993 Megan   28.89 11
Kennedy 1993 Jeff    31.11 12
Kennedy 1993 Taz     30.55 11
Kennedy 1993 George  27.56 11
Kennedy 1993 Heather 38.67 15
Kennedy 1994 Nancy   29.90 26
Kennedy 1994 Rusty   30.55 28
Kennedy 1994 Mimi    37.67 22
Kennedy 1994 J.C.    23.33 27
Kennedy 1994 Clark   27.90 25
Kennedy 1994 Rudy    27.78 23
```

```
Kennedy 1994 Samuel  34.44 18
Kennedy 1994 Forrest 28.89 26
Kennedy 1994 Luther  72.22 24
Kennedy 1994 Trey     6.78 18
Kennedy 1994 Albert  23.33 19
Kennedy 1994 Che-Min 26.66 33
Kennedy 1994 Preston 32.22 23
Kennedy 1994 Larry   40.00 26
Kennedy 1994 Anton   35.99 28
Kennedy 1994 Sid     27.45 25
Kennedy 1994 Will    28.88 21
Kennedy 1994 Morty   34.44 25
;
```

Creating the DIVFMT. and USETYPE. Formats

```
proc format;
   value divfmt 1='New England'
                2='Middle Atlantic'
                3='Mountain'
                4='Pacific';
   value usetype 1='Residential Customers'
                 2='Business Customers';
run;
```

Creating the Univ ODS Document

```
ods document name=univ;

title '100 Obs Sampled from a Normal Distribution';
proc univariate data=distrdata noprint;
  var Normal_x;

  histogram Normal_x /normal(noprint) cbarline=grey name='normal';
run;

  title '100 Obs Sampled from an Exponential Distribution';

proc univariate data=distrdata noprint;
  var Exponential_x;

  histogram /exp(fill l=3) cfill=yellow midpoints=.05 to 5.55 by .25
             name='exp';
run;

  ods document close;
title;

  proc document;
   doc;
   doc name=univ;
```

```
   list/levels=all;

     dir univariate#2\exponential_x\fitteddistributions\exponential;
   list;
   list fitquantiles/details;
run;


  quit;
```

Creating the Employee_Data Data Set

```
options source pagesize=60 linesize=80 nodate;

data employee_data;
   input IdNumber $ 1-4 LastName $ 9-19 FirstName $ 20-29
         City $ 30-42 State $ 43-44 /
         Gender $ 1 JobCode $ 9-11 Salary 20-29 @30 Birth date9.
         @43 Hired date9. HomePhone $ 54-65;
   format birth hired date9.;

           datalines;
1919    Adams      Gerald    Stamford     CT
M       TA2        34376     15SEP48      07JUN75    203/781-1255
1653    Alexander  Susan     Bridgeport   CT
F       ME2        35108     18OCT52      12AUG78    203/675-7715
1400    Apple      Troy      New York     NY
M       ME1        29769     08NOV55      19OCT78    212/586-0808
1350    Arthur     Barbara   New York     NY
F       FA3        32886     03SEP53      01AUG78    718/383-1549
1401    Avery      Jerry     Paterson     NJ
M       TA3        38822     16DEC38      20NOV73    201/732-8787
1499    Barefoot   Joseph    Princeton    NJ
M       ME3        43025     29APR42      10JUN68    201/812-5665
1101    Baucom     Walter    New York     NY
M       SCP        18723     09JUN50      04OCT78    212/586-8060
1333    Blair      Justin    Stamford     CT
M       PT2        88606     02APR49      13FEB69    203/781-1777
1402    Blalock    Ralph     New York     NY
M       TA2        32615     20JAN51      05DEC78    718/384-2849
1479    Bostic     Marie     New York     NY
F       TA3        38785     25DEC56      08OCT77    718/384-8816
1403    Bowden     Earl      Bridgeport   CT
M       ME1        28072     31JAN57      24DEC79    203/675-3434
1739    Boyce      Jonathan  New York     NY
M       PT1        66517     28DEC52      30JAN79    212/587-1247
1658    Bradley    Jeremy    New York     NY
M       SCP        17943     11APR55      03MAR80    212/587-3622
1428    Brady      Christine Stamford     CT
F       PT1        68767     07APR58      19NOV79    203/781-1212
```

```
1782    Brown      Jason     Stamford     CT
M       ME2        35345     07DEC58      25FEB80    203/781-0019
1244    Bryant     Leonard   New York     NY
M       ME2        36925     03SEP51      20JAN76    718/383-3334
1383    Burnette   Thomas    New York     NY
M       BCK        25823     28JAN56      23OCT80    718/384-3569
1574    Cahill     Marshall  New York     NY
M       FA2        28572     30APR48      23DEC80    718/383-2338
1789    Caraway    Davis     New York     NY
M       SCP        18326     28JAN45      14APR66    212/587-9000
1404    Carter     Donald    New York     NY
M       PT2        91376     27FEB41      04JAN68    718/384-2946
1437    Carter     Dorothy   Bridgeport   CT
F       A3         33104     23SEP48      03SEP72    203/675-4117
1639    Carter     Karen     Stamford     CT
F       A3         40260     29JUN45      31JAN72    203/781-8839
1269    Caston     Franklin  Stamford     CT
M       NA1        41690     06MAY60      01DEC80    203/781-3335
1065    Chapman    Neil      New York     NY
M       ME2        35090     29JAN32      10JAN75    718/384-5618
1876    Chin       Jack      New York     NY
M       TA3        39675     23MAY46      30APR73    212/588-5634
1037    Chow       Jane      Stamford     CT
F       TA1        28558     13APR52      16SEP80    203/781-8868
1129    Cook       Brenda    New York     NY
F       ME2        34929     11DEC49      20AUG79    718/383-2313
1988    Cooper     Anthony   New York     NY
M       FA3        32217     03DEC47      21SEP72    212/587-1228
1405    Davidson   Jason     Paterson     NJ
M       SCP        18056     08MAR54      29JAN80    201/732-2323
1430    Dean       Sandra    Bridgeport   CT
F       TA2        32925     03MAR50      30APR75    203/675-1647
1983    Dean       Sharon    New York     NY
F       FA3        33419     03MAR50      30APR75    718/384-1647
1134    Delgado    Maria     Stamford     CT
F       TA2        33462     08MAR57      24DEC76    203/781-1528
1118    Dennis     Roger     New York     NY
M       PT3        111379    19JAN32      21DEC68    718/383-1122
1438    Donaldson  Karen     Stamford     CT
F       TA3        39223     18MAR53      21NOV75    203/781-2229
1125    Dunlap     Donna     New York     NY
F       FA2        28888     11NOV56      14DEC75    718/383-2094
1475    Eaton      Alicia    New York     NY
F       FA2        27787     18DEC49      16JUL78    718/383-2828
1117    Edgerton   Joshua    New York     NY
M       TA3        39771     08JUN51      16AUG80    212/588-1239
1935    Fernandez  Katrina   Bridgeport   CT
F       NA2        51081     31MAR42      19OCT69    203/675-2962
1124    Fields     Diana     White Plains NY
F       FA1        23177     13JUL46      04OCT78    914/455-2998
1422    Fletcher   Marie     Princeton    NJ
F       FA1        22454     07JUN52      09APR79    201/812-0902
1616    Flowers    Annette   New York     NY
F       TA2        34137     04MAR58      07JUN81    718/384-3329
```

```
1406    Foster     Gerald    Bridgeport   CT
M       ME2        35185     11MAR49      20FEB75    203/675-6363
1120    Garcia     Jack      New York     NY
M       ME1        28619     14SEP60      10OCT81    718/384-4930
1094    Gomez      Alan      Bridgeport   CT
M       FA1        22268     05APR58      20APR79    203/675-7181
1389    Gordon     Levi      New York     NY
M       BCK        25028     18JUL47      21AUG78    718/384-9326
1905    Graham     Alvin     New York     NY
M       PT1        65111     19APR60      01JUN80    212/586-8815
1407    Grant      Daniel    Mt. Vernon   NY
M       PT1        68096     26MAR57      21MAR78    914/468-1616
1114    Green      Janice    New York     NY
F       TA2        32928     21SEP57      30JUN75    212/588-1092
;
```

Creating the Energy Data Set

```
data energy;
   length State $2;
   input Region Division state $ Type Expenditures @@;
   datalines;
1 1 ME 1 708  1 1 ME 2 379  1 1 NH 1 597   1 1 NH 2 301
1 1 VT 1 353  1 1 VT 2 188  1 1 MA 1 3264  1 1 MA 2 2498
1 1 RI 1 531  1 1 RI 2 358  1 1 CT 1 2024  1 1 CT 2 1405
1 2 NY 1 8786 1 2 NY 2 7825 1 2 NJ 1 4115  1 2 NJ 2 3558
1 2 PA 1 6478 1 2 PA 2 3695 4 3 MT 1 322   4 3 MT 2 232
4 3 ID 1 392  4 3 ID 2 298  4 3 WY 1 194   4 3 WY 2 184
4 3 CO 1 1215 4 3 CO 2 1173 4 3 NM 1 545   4 3 NM 2 578
4 3 AZ 1 1694 4 3 AZ 2 1448 4 3 UT 1 621   4 3 UT 2 438
4 3 NV 1 493  4 3 NV 2 378  4 4 WA 1 1680  4 4 WA 2 1122
4 4 OR 1 1014 4 4 OR 2 756  4 4 CA 1 10643 4 4 CA 2 10114
4 4 AK 1 349  4 4 AK 2 329  4 4 HI 1 273   4 4 HI 2 298
;
```

Creating the Exprev Data Set

```
data exprev;
input Country $ 1-24 Emp_ID $ 25-32 Order_Date $  Ship_Date $  Sale_Type $ & Quantity Price Cost;

datalines;
Antarctica              99999999    1/1/05    1/7/05          Internet   2    92.60    20.70
Puerto Rico             99999999    1/1/05    1/5/05          Catalog    14   51.20    12.10
Virgin Islands (U.S.)   99999999    1/1/05    1/4/05          In Store   25   31.10    15.65
Aruba                   99999999    1/1/05    1/4/05          Catalog    30   123.70   59.00
Bahamas                 99999999    1/1/05    1/4/05          Catalog    8    113.40   28.45
Bermuda                 99999999    1/1/05    1/4/05          Catalog    7    41.00    9.25
Belize                  120458      1/2/05    1/2/05          In Store   2    146.40   36.70
British Virgin Islands  99999999    1/2/05    1/5/05          Catalog    11   40.20    20.20
```

```
Canada                  99999999    1/2/05      1/5/05          Catalog     100   11.80     5.00
Cayman Islands          120454      1/2/05      1/2/05          In Store    20    71.00     32.30
Costa Rica              99999999    1/2/05      1/6/05          Internet    31    53.00     26.60
Cuba                    121044      1/2/05      1/2/05          Internet    12    42.40     19.35
Dominican Republic      121040      1/2/05      1/2/05          Internet    13    48.00     23.95
El Salvador             99999999    1/2/05      1/6/05          Catalog     21    266.40    66.70
Guatemala               120931      1/2/05      1/2/05          In Store    13    144.40    65.70
Haiti                   121059      1/2/05      1/2/05          Internet    5     47.90     23.45
Honduras                120455      1/2/05      1/2/05          Internet    20    66.40     30.25
Jamaica                 99999999    1/2/05      1/4/05          In Store    23    169.80    38.70
Mexico                  120127      1/2/05      1/2/05          In Store    30    211.80    33.65
Montserrat              120127      1/2/05      1/2/05          In Store    19    184.20    36.90
Nicaragua               120932      1/2/05      1/2/05          Internet    16    122.00    28.75
Panama                  99999999    1/2/05      1/6/05          Internet    20    88.20     38.40
Saint Kitts/Nevis       99999999    1/2/05      1/6/05          Internet    20    41.40     18.00
St. Helena              120360      1/2/05      1/2/05          Internet    19    94.70     47.45
St. Pierre/Miquelon     120842      1/2/05      1/16/05         Internet    16    103.80    47.25
Turks/Caicos Islands    120372      1/2/05      1/2/05          Internet    10    57.70     28.95
United States           120372      1/2/05      1/2/05          Internet    20    88.20     38.40
Anguilla                99999999    1/2/05      1/6/05          In Store    15    233.50    22.25
Antigua/Barbuda         120458      1/2/05      1/2/05          In Store    31    99.60     45.35
Argentina               99999999    1/2/05      1/6/05          In Store    42    408.80    87.15
Barbados                99999999    1/2/05      1/6/05          In Store    26    94.80     42.60
Bolivia                 120127      1/2/05      1/2/05          In Store    26    66.00     16.60
Brazil                  120127      1/2/05      1/2/05          Catalog     12    73.40     18.45
Chile                   120447      1/2/05      1/2/05          In Store    20    19.10     8.75
Colombia                121059      1/2/05      1/2/05          Internet    28    361.40    90.45
Dominica                121043      1/2/05      1/2/05          Internet    35    121.30    57.80
Ecuador                 121042      1/2/05      1/2/05          In Store    11    100.90    50.55
Falkland Islands        120932      1/2/05      1/2/05          In Store    15    61.40     30.80
French Guiana           120935      1/2/05      1/2/05          Catalog     15    96.40     43.85
Grenada                 120931      1/2/05      1/2/05          Catalog     19    56.30     25.05
Guadeloupe              120445      1/2/05      1/2/05          Internet    21    231.60    48.70
Guyana                  120455      1/2/05      1/2/05          In Store    25    132.80    30.25
Martinique              120841      1/2/05      1/3/05          In Store    16    56.30     31.05
Netherlands Antilles    99999999    1/2/05      1/6/05          In Store    31    41.80     19.45
Paraguay                120603      1/2/05      1/2/05          Catalog     17    117.60    58.90
Peru                    120845      1/2/05      1/2/05          Catalog     12    93.80     41.75
St. Lucia               120845      1/2/05      1/2/05          Internet    19    64.30     28.65
Suriname                120538      1/3/05      1/3/05          Internet    22    110.80    29.35
;
run;
```

Creating the Gov Data Set

```
data gov;
   Label Citygovt='City Government Form'
        Robgrp='Number of Citizens Robbed';
   Input Citygovt Robgrp Weight;  Missing N;
   Format Citygovt Govtfmt. Robgrp Robfmt.;
   LOOP: OUTPUT; WEIGHT=WEIGHT-1; IF WEIGHT>0 THEN GOTO LOOP;
   DROP WEIGHT;
datalines;
0 1  6
0 3  3
0 2  7
0 4  5
N N 10
-3 1 47
-3 3 49
-3 2 63
-3 4 52
. 2  1
3 1 31
3 2 37
3 3 27
3 4 55
3 .  1
;
```

Creating the Grain_Production Data Set

```
data grain_production;
   length Country $ 3 Type $ 5;
   input Year country $ type $ Kilotons;
   datalines;
1995 BRZ  Wheat    1516
1995 BRZ  Rice     11236
1995 BRZ  Corn     36276
1995 CHN  Wheat    102207
1995 CHN  Rice     185226
1995 CHN  Corn     112331
1995 IND  Wheat    63007
1995 IND  Rice     122372
1995 IND  Corn     9800
1995 INS  Wheat    .
1995 INS  Rice     49860
1995 INS  Corn     8223
1995 USA  Wheat    59494
1995 USA  Rice     7888
1995 USA  Corn     187300
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 CHN  Wheat    109000
1996 CHN  Rice     190100
1996 CHN  Corn     119350
1996 IND  Wheat    62620
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 INS  Wheat    .
1996 INS  Rice     51165
1996 INS  Corn     8925
1996 USA  Wheat    62099
1996 USA  Rice     7771
1996 USA  Corn     236064
;
```

Creating the Iron Data Set

The data set Iron contains data from Draper and Smith (p. 98).*

```
data iron;
   input Fe Loss @@;
   datalines;
0.01 127.6   0.48 124.0   0.71 110.8   0.95 103.9
1.19 101.5   0.01 130.1   0.48 122.0   1.44  92.3
0.71 113.1   1.96  83.7   0.01 128.0   1.44  91.4
1.96  86.2
;
```

Creating the Model Data Set

```
data one;
   input year import doprod stock consum;
   datalines;
49 15.9 149.3 4.2 108.1
50 16.4 161.2 4.1 114.8
51 19.0 171.5 3.1 123.2
52 19.1 175.5 3.1 126.9
53 18.8 180.8 1.1 132.1
54 20.4 190.7 2.2 137.7
55 22.7 202.1 2.1 146.0
56 26.5 212.4 5.6 154.1
57 28.1 226.1 5.0 162.3
58 27.6 231.9 5.1 164.3
59 26.3 239.0 0.7 167.6
60 31.1 258.0 5.6 176.8
61 33.3 269.8 3.9 186.6
62 37.0 288.4 3.1 199.7
63 43.3 304.5 4.6 213.9
64 49.0 323.4 7.0 223.8
65 50.3 336.8 1.2 232.0
66 56.6 353.9 4.5 242.9
;



data model;
input year 1-2 a 3-9 .3 b 10-17 .3 r4 18-24 .3 r8 25-31 .3
             c 32-38 .3 d 39-45 .3 e 46-51 .3 r23 52-58 .3
             r24 59-64 .3 r29 65-70 .3 r33 71-77 .3 ;
datalines;
60 994534 53552371656049 9362944261250 89214236319711402991060045  8780 335066
611253576 558064317701511067142465093O 99334538746512173601515O7 36871  49192
621318885 62144801893292107568846957361068665450288131729317801 4 66671 566079
```

* Draper, N. and Smith, H. (1998), *Applied Regression Analysis, Second Edition*, New York: John Wiley & Sons.

```
631507969 666125121046261533088511701311673695162821579148179797106485  -4568
641811051 731945021737841454106554095914677245822921945534206255145948 -10940
652532026 816707123363201962785640926221155676314091906268218759195733-145568
661845213 889039326806342223395649307215331186055041732948288322275400 132143
671745867 982910727559092191906712443321301786392551689676279632372882 206952
681408113109029123088034303123479095451531823663475166439633903156093l-197937
69 803331106487483477032288955876371 77997765524616727183686255463771 521929
701234567890123456789012345678901234567890123456789012345678901234567890123456789012345
719876543210987654321098765432109876543210987654321098765432109765432109876547
725432109876543215432109876543210987654321098765432109876543210987654321098765
run;


data model;
set model;
  r4=r4/10;
  r8=r8/10;
  d=d/10;
  e=e/10;
  r23=r23/10;
  r33=r33/10;
  a=a/10;
  b=b/10;
  c=c/10;
  r24=r24/10;
  r29=r29/10;
run;
```

Creating the Plants Data Set

```
data plants;
   input type $ @;
   do block=1 to 3;
      input stemleng @;
      output;
   end;
   datalines;
   clarion  32.7 32.3 31.5
   clinton  32.1 29.7 29.1
   knox     35.7 35.9 33.1
   o'neill  36.0 34.2 31.2
   compost  31.8 28.0 29.2
   wabash   38.2 37.8 31.9
   webster  32.5 31.1 29.7
   ;
run;
```

Creating the Plant_Stat Data Set

```
data plant_stats;
  do month = 1 to 12;
```

```
    age  = 2 + 0.3*rannor(345467);
    age2 = 3 + 0.3*rannor(345467);
    age3 = 4 + 0.4*rannor(345467);
    output;
  end;
run;
```

Creating the StatePop Data Set

```
data statepop;
   input State $ CityPop_80 CityPop_90
         NonCityPop_80 NonCityPop_90 Region;
   format region 1.;
   label citypop_80=   '1980 metropolitan pop in millions'
         noncitypop_80='1980 nonmetropolitan pop in millions'
         citypop_90=   '1990 metropolitan pop in millions'
         noncitypop_90='1990 nonmetropolitan pop in million'
         region='Geographic region';
  datalines;
ME    .405    .443   .721   .785  1
NH    .535    .659   .386   .450  1
VT    .133    .152   .378   .411  1
MA   5.530   5.788   .207   .229  1
RI    .886    .938   .061   .065  1
CT   2.982   3.148   .126   .140  1
NY  16.144  16.515  1.414  1.475  1
NJ   7.365   7.730    .A     .A   1
PA  10.067  10.083  1.798  1.799  1
DE    .496    .553   .098   .113  2
MD   3.920   4.439   .297   .343  2
DC    .638    .607      .      .  2
VA   3.966   4.773  1.381  1.414  2
WV    .796    .748  1.155  1.045  2
NC   3.749   4.376  2.131  2.253  2
SC   2.114   2.423  1.006  1.064  2
GA   3.507   4.352  1.956  2.127  2
FL   9.039  12.023   .708   .915  2
KY   1.735   1.780  1.925  1.906  2
TN   3.045   3.298  1.546  1.579  2
AL   2.560   2.710  1.334  1.331  2
MS    .716    .776  1.805  1.798  2
AR    .963   1.040  1.323  1.311  2
LA   3.125   3.160  1.082  1.060  2
OK   1.724   1.870  1.301  1.276  2
TX  11.539  14.166  2.686  2.821  2
OH   8.791   8.826  2.007  2.021  3
IN   3.885   3.962  1.605  1.582  3
IL   9.461   9.574  1.967  1.857  3
MI   7.719   7.698  1.543  1.598  3
WI   3.176   3.331  1.530  1.561  3
MN   2.674   3.011  1.402  1.364  3
```

```
IA   1.198   1.200  1.716  1.577  3
MO   3.314   3.491  1.603  1.626  3
ND    .234    .257   .418   .381  3
SD    .194    .221   .497   .475  3
NE    .728    .787   .842   .791  3
KS   1.184   1.333  1.180  1.145  3
MT    .189    .191   .598   .608  4
ID    .257    .296   .687   .711  4
WY    .141    .134   .329   .319  4
CO   2.326   2.686   .563   .608  4
NM    .675    .842   .628   .673  4
AZ   2.264   3.106   .453   .559  4
UT   1.128   1.336   .333   .387  4
NV    .666   1.014   .135   .183  4
WA   3.366   4.036   .776   .830  4
OR   1.799   1.985   .834   .858  4
CA  22.907  28.799   .760   .961  4
AK    .174    .226   .227   .324  4
HI    .763    .836   .202   .272  4
;
```

Creating the Table1 Table Definition

```
proc template;
   define table table1;
     mvar sysdate9;
     dynamic colhd;
     classlevels=on;

     define column char_var;
      generic=on;
      blank_dups=on;
      header=colhd;
      style=cellcontents;
   end;

     define column num_var;
      generic=on;
      header=colhd;
      style=cellcontents;
   end;

     define footer table_footer;
      text 'Prepared on ' sysdate9;
   end;

     end;
run;
```

Programs That Illustrate Inheritance

The programs in this section show the PROC TEMPLATE steps that were used in "Understanding Styles, Style Elements, and Style Attributes" on page 540 to illustrate inheritance in style definitions. These programs also show the SAS code that uses the style definitions.

Using the FROM option

This program generates the HTML output in the section "Using the FROM Option" on page 546.

- This version of the code uses the FROM option in the STYLE statement to create the Colors style element in the Concepts.Style2 style definition.

```
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
   datalines;
Brazil        Rice   10035
China         Rice   190100
India         Rice   120012
Indonesia     Rice   51165
United States Rice   7771
;

proc template;
   define table mytable;
      column x y z w;
      define x;
         style=celldatasimple;
         dataname=country;
         header='Country';
      end;
      define y;
           style=celldataemphasis;
           dataname=grain;
           header='Grain';
      end;
      define z;
         style=celldatalarge;
         dataname=kilotons;
         header='Kilotons';
      end;
      define w;
         style=celldatasmall;
         dataname=kilotons;
         header='Kilotons';
      end;
   end;
run;
```

```
proc template;
   /* to ensure a fresh start with the styles */
   delete concepts.style1;
   delete concepts.style2;
run;

proc template;
   define style concepts.style1;
      style colors /
         'default'=white
         'fancy'=very light vivid blue
         'medium'=red ;
      style celldatasimple /
         fontfamily=arial
         backgroundcolor=colors('fancy')
         color=colors('default');
      style celldataemphasis from celldatasimple /
         color=colors('medium')
         fontstyle=italic;
      style celldatalarge from celldataemphasis /
         fontweight=bold
         fontsize=3;
   end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
      style colors from colors/
         'dark'=dark blue;
      style celldataemphasis from celldataemphasis /
         backgroundcolor=white;
      style celldatasmall from celldatalarge /
         fontsize=5
         color=colors('dark')
         backgroundcolor=colors('medium');
   end;
run;
ods html body='display1-body.htm'
         style=concepts.style2;
data _null_;
   set test;
   file print ods=(template='mytable');
   put _ods_;
run;
ods html close;
```

□ This version of the code does not use the FROM option in the STYLE statement to create the Colors style element in the Concepts.Style2 style definition.

```
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
```

```
   datalines;
Brazil        Rice   10035
China         Rice   190100
India         Rice   120012
Indonesia     Rice   51165
United States Rice   7771
;

proc template;
   define table mytable;
      column x y z w;
      define x;
         style=celldatasimple;
         dataname=country;
         header='Country';
      end;
      define y;
           style=celldataemphasis;
           dataname=grain;
           header='Grain';
      end;
      define z;
         style=celldatalarge;
         dataname=kilotons;
         header='Kilotons';
      end;
      define w;
         style=celldatasmall;
         dataname=kilotons;
         header='Kilotons';
      end;
   end;
run;

proc template;
   /* to ensure a fresh start with the styles */
   delete concepts.style1;
   delete concepts.style2;
run;

proc template;
   define style concepts.style1;
      style colors /
         'default'=white
         'fancy'=very light vivid blue
         'medium'=red ;
      style celldatasimple /
         fontfamily=arial
         backgroundcolor=colors('fancy')
         color=colors('default');
      style celldataemphasis from celldatasimple /
         color=colors('medium')
         fontstyle=italic;
      style celldatalarge from celldataemphasis /
```

```
         fontweight=bold
         fontsize=3;
   end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
      style colors /
         'dark'=dark blue;
      style celldataemphasis from celldataemphasis /
         backgroundcolor=white;
      style celldatasmall from celldatalarge /
         fontsize=5
         color=colors('dark')
         backgroundcolor=colors('medium');
   end;
run;
ods html body='display1-body.htm'
         style=concepts.style2;
data _null_;
   set test;
   file print ods=(template='mytable');
   put _ods_;
run;
ods html close;
```

Inheritance Compatibility Across SAS Versions

This program generates the HTML output in the section "Inheritance Compatibility across Versions" on page 548.

□ This version of the code uses SAS 9.2 names for tyle attributes supplied by SAS.

```
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
   datalines;
Brazil        Rice   10035
China         Rice   190100
India         Rice   120012
Indonesia     Rice   51165
United States Rice   7771
;

proc template;
   define table mytable;
      column x y z w;
      define x;
         style=celldatasimple;
         dataname=country;
         header='Country';
      end;
```

```
      define y;
           style=celldataemphasis;
           dataname=grain;
           header='Grain';
      end;
      define z;
         style=celldatalarge;
         dataname=kilotons;
         header='Kilotons';
      end;
      define w;
         style=celldatasmall;
         dataname=kilotons;
         header='Kilotons';
      end;
   end;
run;

proc template;
   /* to ensure a fresh start with the styles */
   delete concepts.style1;
   delete concepts.style2;
run;

proc template;
   define style concepts.style1;
      style celldatasimple /
         fontfamily=arial
         backgroundcolor=very light vivid blue
         color=white;
      style celldataemphasis from celldatasimple /
         color=red
         fontstyle=italic;
      style celldatalarge from celldataemphasis /
         fontweight=bold
         fontsize=5;
   end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
      style celldataemphasis from celldataemphasis /
         backgroundcolor=yellow;
      style celldatasmall from celldatalarge /
         fontsize=2;
   end;

ods html body='display1-body.htm'
         style=concepts.style2;
data _null_;
   set test;
   file print ods=(template='mytable');
   put _ods_;
```

```
run;
ods html close;
```

□ This version of the code uses SAS 9.1 names for style attributes that are supplied by SAS.

```
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
   datalines;
Brazil        Rice   10035
China         Rice   190100
India         Rice   120012
Indonesia     Rice   51165
United States Rice   7771
;

proc template;
   define table mytable;
      column x y z w;
      define x;
         style=celldatasimple;
         dataname=country;
         header='Country';
      end;
      define y;
           style=celldataemphasis;
           dataname=grain;
           header='Grain';
      end;
      define z;
         style=celldatalarge;
         dataname=kilotons;
         header='Kilotons';
      end;
      define w;
         style=celldatasmall;
         dataname=kilotons;
         header='Kilotons';
      end;
   end;
run;

proc template;
   /* to ensure a fresh start with the styles */
   delete concepts.style1;
   delete concepts.style2;
run;

proc template;
   define style concepts.style1;
      style celldatasimple /
         fontface=arial
```

```
            background=very light vivid blue
            foreground=white;
         style celldataemphasis from celldatasimple /
            foreground=red
            fontstyle=italic;
         style celldatalarge from celldataemphasis /
            fontweight=bold
            fontsize=5;
      end;
   run;

   proc template;
      define style concepts.style2;
         parent=concepts.style1;
         style celldataemphasis from celldataemphasis /
            background=yellow;
         style celldatasmall from celldatalarge /
            fontsize=2;
      end;

   ods html body='display1-body.htm'
            style=concepts.style2;
   data _null_;
      set test;
      file print ods=(template='mytable');
      put _ods_;
   run;
   ods html close;
```

Creating the Nlits Data Set

```
/* The mixed_case lets you use an nlit in the data set name.   */
%libcat(nlits, pathname=nlits, opt=mixed_case=yes);

data nlits.'          Stats'n;
   input Price Quantity City  $;
   datalines;
3750  150 Brazil
5000  200 Canada
10250 410 France
;

data nlits.'     Stats2'n;
   input Price Quantity City $;
   datalines;
3750  150 Brazil
5000  200 Canada
10250 410 France
;
```

APPENDIX

2

ODS and the HTML Destination

HTML Links and References Produced by the HTML Destination

What Are Links and References?

An HTML link is a place in a document that enables you to jump to another specific place in the same document or in another document. A browser typically highlights the text that is between the tags that begin and end the link. When you click on the highlighted text, the browser displays the text at the link target. The browser might then display the contents of the target in the active window, or it might open another browser window that displays the contents of the target.

An HTML reference names a file for the browser to display. When a browser reads a reference, it displays the referenced file as if it were part of the file that it is displaying. You can't tell by looking at the browser's display that some of the material is in the file that you are actually viewing and that some is referenced.

When you use ODS, the software automatically creates the links and references that you need. You can, however, customize these links to some extent. If you want to do so, then you will need to understand how HTML implements links and references.

Implementing HTML Links and References

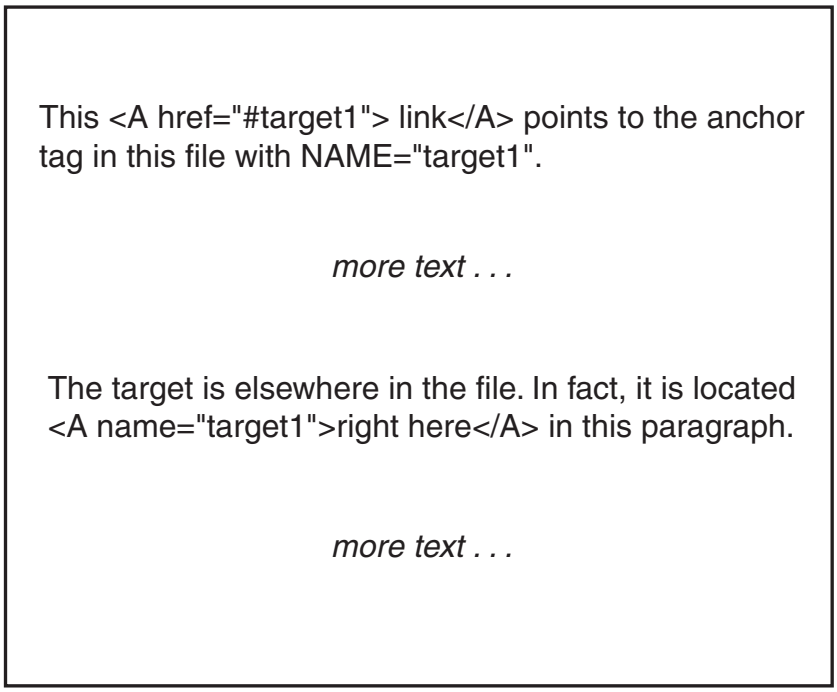
Note: This simplified discussion of HTML links and references is designed to provide information that will help you understand what ODS does when it builds links and references for you. For a complete discussion of HTML tagging, consult one of the many reference books that are available on the subject. △

Each link in HTML is implemented with a combination of two sets of **<A>** (anchor) tags. One anchor tag, which is the starting point of the link, has an **HREF** attribute that

identifies the anchor tag to link to. The other anchor tag, which is the target of the link, has a **NAME** attribute. This **NAME** attribute is what the **HREF** attribute in the first anchor tag points to. The value of each **NAME** attribute in a file must be unique so that each value of **HREF** points to a single, unambiguous location. The following figure illustrates linking within a file:

Figure A2.1 Linking within a File

The browser highlights the word **link**. When you click on **link**, the browser positions the target **right here** in the active window.



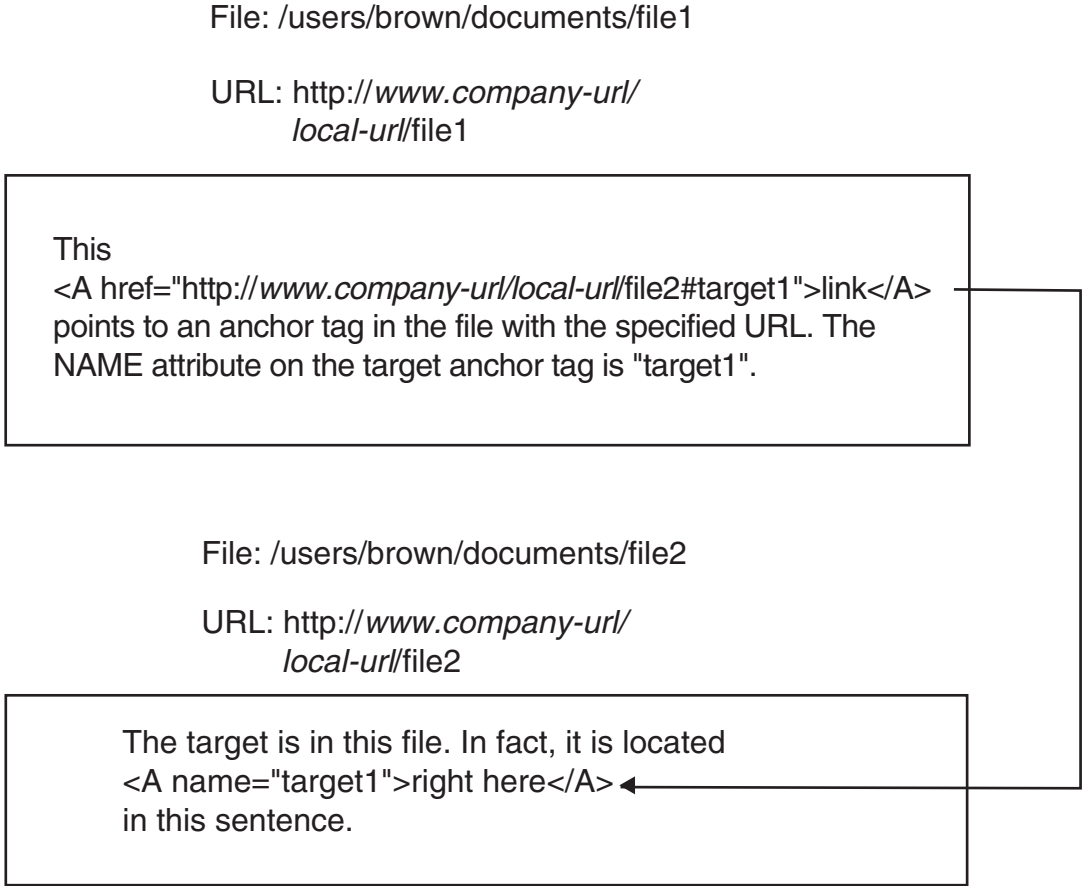
The important features at the starting point of this link are

- The **<A>** and **</A>** tags surround the text that the browser will highlight.
- The **HREF** attribute points to the link's target. The target is an anchor tag whose **NAME** attribute matches the text that follows the pound sign in the **HREF** attribute. Because no text precedes the pound sign (#), the browser knows that the target is in the same file as the anchor.

When a link points to a target outside the file that is being displayed, the **HREF** attribute must include the path to that file. The path can be the path within the file system or the uniform resource locator (URL) of the file. The following figure illustrates a link from one file to another file that is specified with a URL:

Figure A2.2 Linking to Another File

The browser highlights the word **link**. When you click on **link**, the browser positions the target **right here** in the active window or opens another window that displays the target.



The important features at the starting point (the anchor) of the link are

- The **<A>** and **</A>** tags surround the text that the browser will highlight.
- The **HREF** attribute points to the link's target. The text that precedes the pound sign (#) identifies the file that contains the target.

ODS provides features that enable you to customize the text that precedes the pound sign and the text that follows the pound sign. For information on how to do this, see the discussions of *file-specification*, ANCHOR=, BASE=, PATH=, and GPATH= in the "ODS HTML Statement" on page 124 as well as "How ODS Constructs Links and References" on page 894.

HTML implements references in much the same way as it implements links. The main difference is that a link points to a particular location within a file and that a reference points to the file itself. HTML uses the **SRC** attribute to identify a file to reference. The value of the **SRC** attribute is constructed the same way that the value of the **HREF** attribute is constructed except that there is no pound sign and no text following it.

How ODS Constructs Links and References

Several options in the ODS HTML statement affect how ODS constructs the links and references that point from the frame to the table of contents, table of pages, and body file and from the table of contents or table of pages to the body file. Links are made as **HREF** attributes on **<A>** (anchor) tags inside the HTML files. Each **HREF** attribute points to the **NAME** attribute on another **<A>** tag. The **HREF** must identify both the file that contains the target and the name of the anchor within that file. The value of **HREF** must be a valid target in a valid URL. It uses the following form:

<A href="*URL#anchor-name*">

ODS constructs the value of an **HREF** attribute based on information that you provide in the ODS HTML statement.

Note: HTML references to files use other tags, but the logic for creating the string that identifies the file is the same as the logic for creating an **HREF** attribute (see "How ODS Constructs Links and References" on page 894). △

The URL in an **HREF** attribute includes information from three options in the ODS HTML statement: the BASE option; the GPATH= or the PATH= option; and the BODY=, the CONTENTS=, or the PAGE= option.

1. If you specify BASE=, then the value of that option is the first part of the URL for every **HREF** attribute that ODS writes.
2. If you specify GPATH= or PATH=, then the next part of the URL in an **HREF** attribute comes from that option.

   If the file that you are linking to is a high-resolution graphic, then ODS uses information from the GPATH= option as the next part of the **HREF**. For information on these options, see the discussion of GPATH= and the discussion of PATH= in the "ODS HTML Statement" on page 124. The following table shows how ODS uses information from the GPATH= option in the URL in **HREF** attributes:

Table A2.1 Building an HREF Attribute from the GPATH= Option

file-specification **in GPATH=**	**URL= Suboption**	**Information ODS Uses in the Second Part of the URL in the HREF attribute***
An *external-file* or *libref.catalog*	Not specified	The name of the file
An *external-file* or *libref.catalog*	Specified, but not NONE	The value of the URL= suboption
An *external-file* or *libref.catalog*	NONE	No information from GPATH=
A *fileref*	Specified or not specified	No information from GPATH=

* If you do not specify GPATH=, then ODS uses the value of PATH= to create this part of the **HREF**.

If the file that you are linking to is not a high-resolution graphic, then ODS uses information from the PATH= option as the next part of the **HREF**. The following table shows how ODS uses information from the PATH= option in the URL in **HREF** attributes:

Table A2.2 Building an HREF Attribute from the PATH= Option

file-specification	**URL= Suboption**	**Information Used in the Second Part of the URL in the HREF Attribute**
external-file or *libref.catalog*	Not specified	The name of the file
external-file or *libref.catalog*	Specified, but not NONE	The value of the URL= suboption
external-file or *libref.catalog*	NONE	No information from PATH=
fileref	Specified or not specified	No information from PATH=

Note: If you use a fileref as the file specification in the BODY=, CONTENTS=, or PAGE= option in the ODS HTML statement, and you do not use the URL= suboption in that option, then ODS does not use information from GPATH= or PATH= when it creates the complete URL for any corresponding **HREF** attributes. △

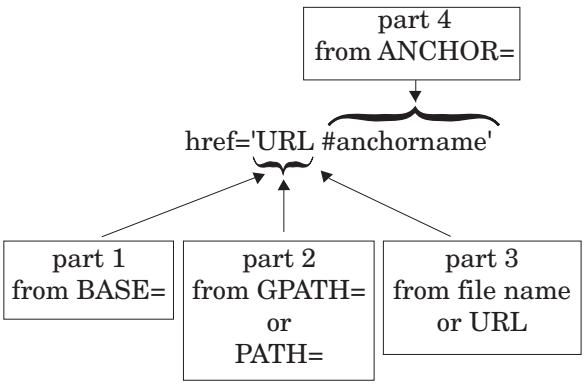
3 The last part of the URL that is used in an **HREF** attribute is, by default, the name of the file that contains the target. ODS determines the name of the file from the *file-specification* that you use in the BODY=, CONTENTS=, or PAGE= option. (ODS does not create links or references to frame files.) For more information on these options, see the discussion of file-specification on page 158.

If you specify the URL= suboption in one of these options, then ODS uses the string that you specify instead of the filename.

Note: If you use a fileref as the file specification and do not use the URL= suboption, then ODS does not use information from GPATH= or PATH= when it creates the complete URL for the **HREF** attribute. △

The *anchor-name* comes from the value of the ANCHOR= option.
The following figure illustrates the creation of the **HREF**:

Figure A2.3 Creating the Value of an HREF Attribute



Files Produced by the HTML Destination

The HTML destination can produce four kinds of files: body, contents, frame, and page files. You create these files with options in the ODS HTML statement (see "ODS HTML Statement" on page 124 for details).

The Body File

The body file contains HTML output that is generated from the output objects that your SAS job creates. The style and the table template that the job uses determine the appearance and content of the tables and the cells within them.

Typically, when you route an output object that does not contain graphics to the HTML destination, ODS places the results within **<TABLE>** tags, generating them as one or more HTML tables.

Graphics output is produced according to the SAS code that generates it. Instead of using **<TABLE>** tags, the body file contains an **<IMG>** (image) tag that references the graphic. When you view the body file in a browser, you cannot tell that the graphic is not part of the body file because the **<IMG>** tag displays it in the browser.

Note: Very few procedures produce output objects that are neither tabular nor graphics. In these cases, the output is not tagged as an HTML table. △

Titles and footnotes in the body file are generated as HTML tables of their own near the top and bottom of each page of HTML output.

Note: For graphics output, titles and footnotes are, by default, part of the graphics file. You can use the NOGTITLE and NOGFOOTNOTE options to place them in the body file instead. See the discussion of GTITLE and GFOOTNOTE in "ODS HTML Statement" on page 124 for more information. △

All **<TABLE>** tags and all **<IMG>** tags are potential targets for links or references (see "How ODS Constructs Links and References" on page 894). Therefore, ODS must provide an **<A>** tag with a **NAME** attribute close to each **<TABLE>** and **<IMG>** tag for links and references to point to. The **NAME** attribute on the anchor tag becomes the final part of any reference or link to the table. ODS inserts anchor tags in its HTML output as follows:

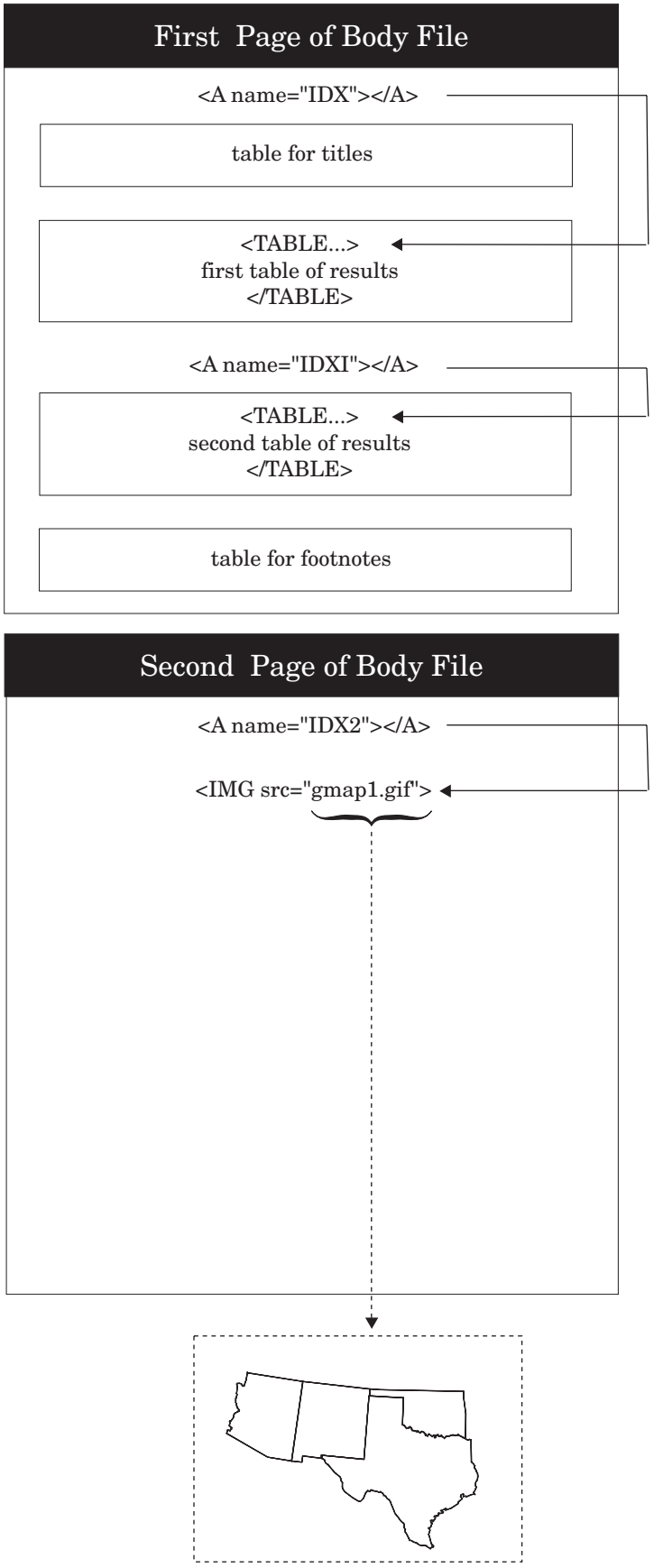
- ODS places an anchor tag near the top of each page, before all tables on the page (including the table that holds the titles) and before all images. This anchor is the target for links to the first table (excluding any titles) or to the first image on the page.

  Note: Each procedure or DATA step starts a new page. In addition, ODS produces a new page of output whenever the SAS program explicitly asks for a new page. For example, if you use the page dimension in PROC TABULATE, then you create a page for each value of the variable that defines the pages. In this context, the word "page" has nothing to do with the PAGESIZE= setting in your SAS session. △
- ODS places an anchor tag slightly before each **<TABLE>** tag, provided that the table contains results (not titles or footnotes) and that it is not the first table or image on the page.
- ODS places an anchor tag slightly before each **<IMG>** tag, provided that it is not the first table or image on a page.

The following figure illustrates the placement of anchor tags from a SAS job that executes two procedures. The first procedure creates two HTML tables of results on a

single page. The page also includes an HTML table for the title and one for the footnote. Solid arrows indicate which **<A>** tag ODS uses as a target for each table. The second procedure creates a GIF file. The titles for this procedure are part of the GIF file (the default behavior). Again, the solid arrow indicates which anchor tag ODS uses as a target when it creates a link to the image. The dashed arrow points to the file that the **<IMG>** tag references.

Figure A2.4 Placement of <A> (anchor) Tags in HTML Output



For a view of this same file through a browser, see Display A2.1 on page 901.

The Contents File

The contents file contains a link to the body file for each HTML table that ODS creates from procedure or DATA step results. The targets for these links are the values of the **NAME** attributes on the anchor tags that are in the body file (see "The Body File" on page 896). For example, an anchor tag that links to the second HTML table of results in Figure A2.4 on page 898 looks like this:

```
<A href="pop-body.htm#IDX1">
```

In this anchor tag,

- pop-body.htm identifies the file that contains the target.
- #IDX1 provides the name of the target.

You can view the contents file directly in the browser, or, if you make a frame file, you can see the contents file as part of the frame file (see "The Frame File" on page 899).

The Page File

The page file contains a link to the body file for each page of HTML output that ODS creates from procedure or DATA step results. The targets for these links are the values of the **NAME** attributes on the anchor tags that are in the body file (see "The Body File" on page 896). For example, an anchor tag that links to the second page of results in Figure A2.4 on page 898 looks like this:

```
<A href="pop-body.htm#IDX2">
```

In this anchor tag,

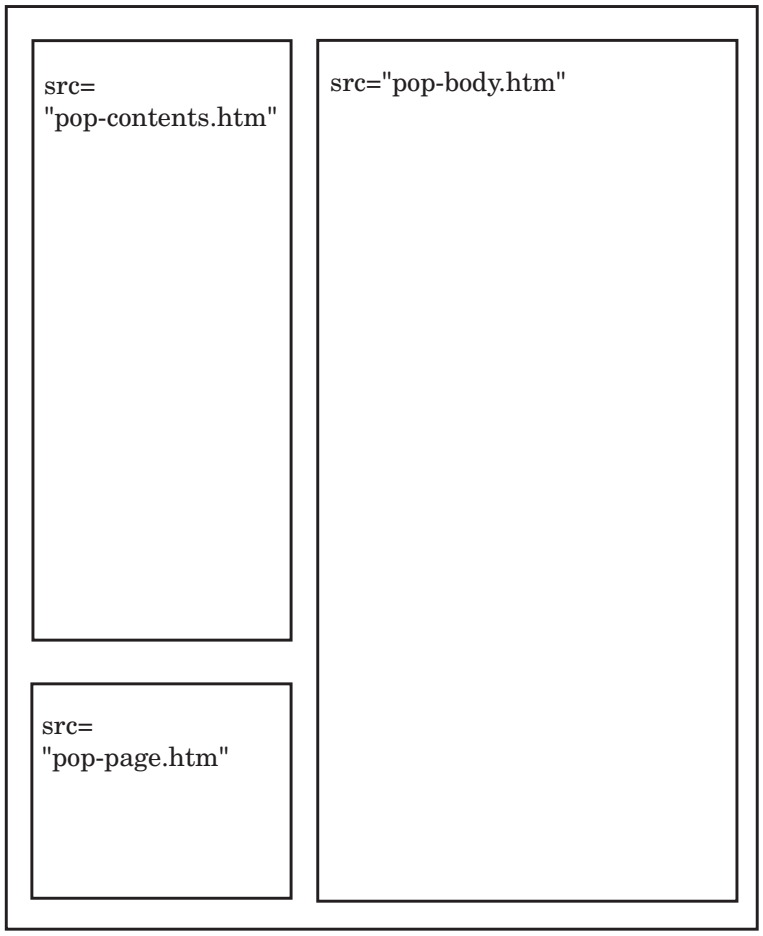
- pop-body.htm identifies the file that contains the target.
- #IDX2 provides the name of the target.

You can view the page file directly in the browser, or, if you make a frame file, you can see the page file as part of the frame file (see "The Frame File" on page 899).

The Frame File

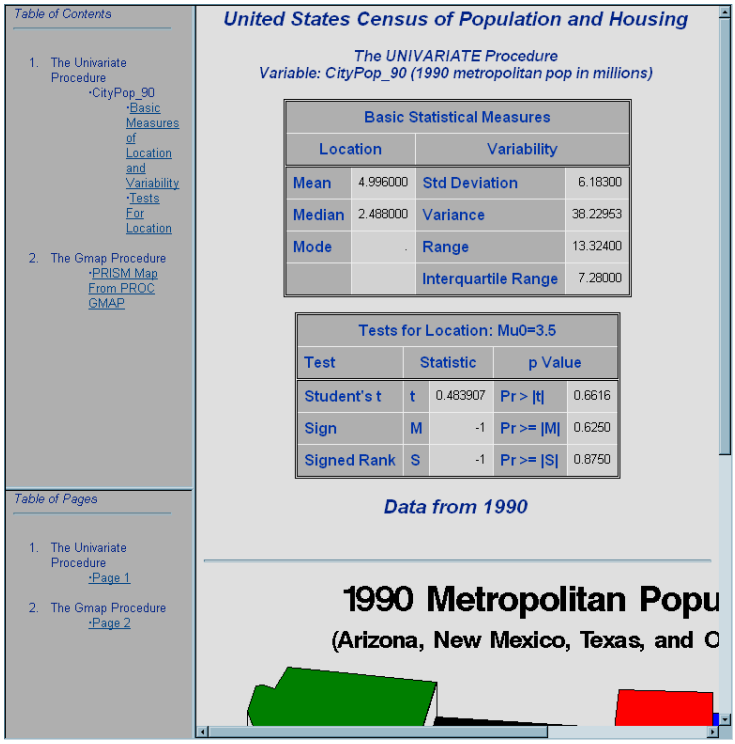
The frame file provides a simultaneous view of the body file and the contents file, the page file, or both. The following figure illustrates how a frame that references both the contents and page files looks (in part) to an ASCII editor. The **SRC** attribute identifies a file to display in the browser. ODS constructs the value for the **SRC** attribute the same way that it constructs the value for an **HREF** attribute in a page or contents file (see Schematic of an HTML Frame File on page 900).

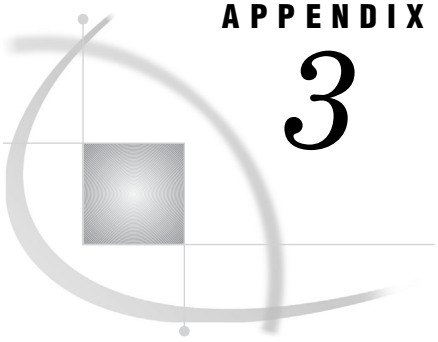
Figure A2.5 Schematic of an HTML Frame File



Display A2.1 on page 901 shows the same frame file viewed from a browser.

Display A2.1 Browser View of HTML Frame File





APPENDIX

3 ODS HTML Statements for Running Examples in Different Operating Environments

Using a z/OS UNIX System Services HFS Directory for HTML Output

```
/* Specify the files to create for the HTML output. */
/* The PATH= option specifies the location for all  */
/* the HTML files. The URL= suboption prevents      */
/* information from PATH= from appearing in the     */
/* links and references that ODS creates. The URLs  */
/* will be the same as the file specifications.     */
ods html body='odsexample-body.htm'
         contents='odsexample-contents.htm'
         page='odsexample-page.htm'
         frame='odsexample-frame.htm'
         path='~'(url=none);
```

Using a z/OS PDSE for EBCDIC HTML Output

```
/* Allocate a PDSE for the HTML Output. */
filename pdsehtml '.example.htm'
                  dsntype=library dsorg=po
                  disp=(new, catlg, delete);

/* Specify the files to create for the HTML output. */
/* These files are PDSE members.                    */
/* The PATH= option specifies the location for all  */
/* the HTML files. The URL= suboption prevents      */
/* information from PATH= from appearing in the     */
/* links and references that ODS creates. The URLs  */
/* will be the same as the file specifications.     */
/* The RS= option creates HTML that you can work    */
/* with in an editor and use on a z/OS Web server.  */
```

```
ods html body='odsexb'
         contents='odsexc'
         page='odsexp'
         frame='odsexf'
         path='.example.htm'(url=none)
         rs=none;
```

Using a z/OS PDSE for ASCII HTML Output

```
/* Allocate a PDSE for the HTML Output. */
filename pdsehtml '.example.htm'
                  dsntype=library dsorg=po
                  disp=(new, catlg, delete);

/* Specify the files to create for the HTML output. */
/* These files are PDSE members.                    */
/* The URL= suboption in the HTML-file              */
/* specifications provides a URL that will be valid */
/* after the PDSE members have been moved to an     */
/* ASCII file system. When the files are            */
/* transferred, they must retain their member names */
/* and have the ".htm" extension added in order for */
/* these URLs to be correct.                        */
/* The PATH= option specifies the location for all  */
/* the HTML files. The URL= suboption in the PATH=  */
/* option prevents information from PATH= from      */
/* appearing in the links and references that ODS   */
/* creates because it will not be a valid URL for   */
/* the ASCII file system.                           */
/* The TRANTAB= option creates ASCII HTML that      */
/* you can send to an ASCII-based Web server.       */

ods html body='odsexb' (url='odsexb.htm')
         contents='odsexc' (url='odsexc.htm')
         page='odsexp' (url='odsexp.htm')
         frame='odsexf'
         path='.example.htm'(url=none)
         trantab=ascii;
```

Note: Use a binary transfer to move the files to the Web server. △

APPENDIX

4

ODS Style Elements

General ODS Style Elements

The following table lists all the style elements available for ODS style definitions. The table provides a brief description of each style element and indicates the style elements from which it inherits its attributes. An abstract style element is one that is not used to generate any style element but provides a parent for one or more style elements to inherit.

Table A4.1 Miscellaneous Style Elements

Style Element	Description	Inherits from
Miscellaneous		
Container *	Controls all container-oriented elements	
Continued	Controls continued flag when a table breaks across a page (paginated destinations only)	TitlesAndFooters
ExtendedPage	Message when page won't fit (Printer only)	TitlesAndFooters
PageNo	Controls page numbers for paginated destinations	TitlesAndFooters
Parskip	Controls space between tables	TitlesAndFooters
PrePage	Controls the ODS RTF/ MEASURED PREPAGE= style	
StartUpFunction	This is a Javascript function that is added to the HTML output. Any Javascript code in the TAGATTR= attribute is executed when the page is loaded.	

Style Element	Description	Inherits from
Miscellaneous		
ShutDownFunction	Controls the Shut-Down function. This is a Javascript function that is added to the HTML output. Any Javascript code in the TAGATTR= attribute is executed when the page is exited.	
UserText	Controls the ODS TEXT= style	Note

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.2 Style Elements Affecting Documents

Style Element	Description	Inherits from
Documents		
Document	Controls the various document bodies. This generally includes things like the page background color and page margins.	Container *
Body	Controls the Body file	Document
Frame	Controls the Frame file for HTML	Document
Contents	Controls the Contents file	Document
Pages	Controls the Page file	Document

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.3 Style Elements Affecting Dates

Style Element	Description	Inherits from
Dates		
BodyDate	Controls the date field in the Contents file	ContentsDate
Date	Controls how date fields look	Container *
PagesDate	Controls the date field in the Pages file	Date

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.4 Style Elements Affecting Table of Contents and Table of Pages

Style Element	Description	Inherits from
Table of Contents and Table of Pages		
IndexItem	Controls list items and folders for Contents and Pages	Container *
ContentFolder	Controls the folders in the Contents file	IndexItem
ByContentFolder	Controls the byline folders in the Contents file	ContentFolder
ContentItem	Controls the items in the Contents file	IndexItem
PagesItem	Controls the items in the Pages file	IndexItem
Index	Controls miscellaneous Contents and Pages components	Container *
IndexProcName	Controls the PROC name in the Contents and Pages files	Index *
ContentProcName	Controls the PROC name in the Contents file	IndexProcName
ContentProcLabel	Controls the PROC label in the Contents file	ContentProcName
PagesProcName	Controls the PROC name in the Pages file	IndexProcName
PagesProcLabel	Controls the PROC label in the Pages file	PagesProcName
IndexAction	Determines what happens on mouse-over events for folders and items (HTML only)	IndexItem
FolderAction	Determines what happens on mouse-over events for folders (HTML only)	IndexAction
IndexTitle	Controls the title of Contents and Pages files	Index *
ContentTitle	Controls the title of the Contents file. In styles.default this element contains a PRETEXT= element that print the text “Table of Contents”.	IndexTitle

Table A4.5 Style Elements Affecting Titles and Footers

Style Element	Description	Inherits from
System Titles and Footers		
SysTitleAndFooterContainer	Controls container for system page title and system page footer. This element is usually used to add borders around a title.	Container
TitlesAndFooters	Controls system page title text and system page footer text	Container *
SystemTitle	Controls system title text	TitlesAndFooters
SystemTitle2	Controls system title2 text	SystemTitle
SystemTitle3	Controls system title3 text	SystemTitle2
SystemTitle4	Controls system title4 text	SystemTitle3
SystemTitle5	Controls system title5 text	SystemTitle4
SystemTitle6	Controls system title6 text	SystemTitle5
SystemTitle7	Controls system title7 text	SystemTitle6
SystemTitle8	Controls system title8 text	SystemTitle7
SystemTitle9	Controls system title9 text	SystemTitle8
SystemTitle10	Controls system title10 text	SystemTitle9
SystemFooter	Controls system footer text	TitlesAndFooters
SystemFooter2	Controls system title2 text	SystemFooter
SystemFooter3	Controls system title3 text	SystemFooter2
SystemFooter4	Controls system title4 text	SystemFooter3
SystemFooter5	Controls system title5 text	SystemFooter4
SystemFooter6	Controls system title6 text	SystemFooter5
SystemFooter7	Controls system title7 text	SystemFooter6
SystemFooter8	Controls system title8 text	SystemFooter7
SystemFooter8	Controls system title8 text	SystemFooter7
SystemFooter9	Controls system title9 text	SystemFooter8
SystemFooter10	Controls system title10 text	SystemFooter9

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.6 Style Elements Affecting Procedure Titles

Style Element	Description	Inherits from
PROC Titles		
TitleAndNoteContainer	Controls container for procedure-defined titles and notes	Container
ProcTitle	Controls procedure title text	TitlesAndFooters
ProcTitleFixed	Controls procedure title text that requests a fixed font	ProcTitle

Table A4.7 Style Elements Affecting Bylines

Style Element	Description	Inherits from
Bylines		
BylineContainer	Controls container for the byline. This is generally used to add borders to a byline.	Container
Byline	Controls byline text	TitlesAndFooters

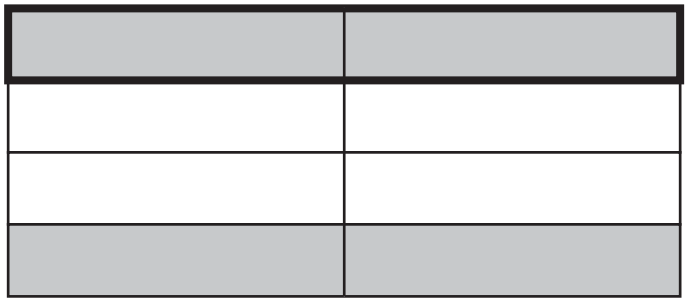
Table A4.8 Style Elements Affecting Notes, Warnings, and Errors

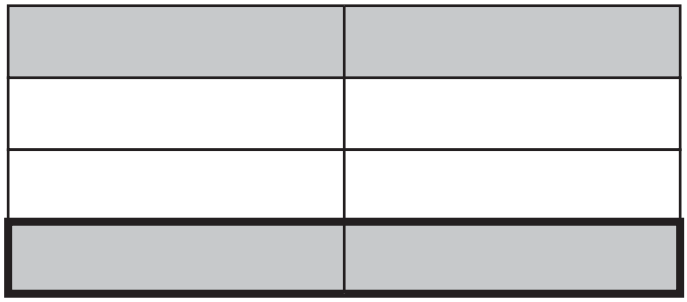
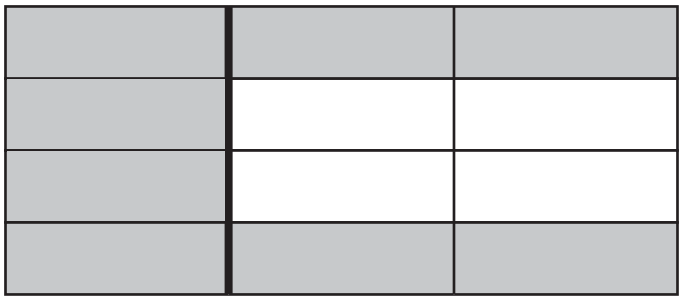
Style Element	Description	Inherits from
Notes, Warnings, and Errors		
Notes, warnings, and errors consist of two pieces: a banner area and a content area as shown in the diagram below. The *Banner elements generally print the content of the banner (that is, "NOTE:", "WARNING:", and so on) using the PRETEXT= attribute.		
		
Note	Controls the container for note banners and note contents	Container *
NoteBanner	Controls the banner for NOTE:s	Note
NoteContent	Controls the contents for NOTE:s	Note
NoteContentFixed	Controls the contents for NOTE:s. Fixed font.	NoteContent
WarnBanner	Controls the banner for WARNING:s	Note
WarnContent	Controls the contents of WARNING:s	Note
WarnContentFixed	Controls the contents for WARNING:s. Fixed font.	WarnContent
ErrorBanner	Controls the banner for ERROR:s	Note

Style Element	Description	Inherits from
Notes, Warnings, and Errors		
ErrorContent	Controls the contents of ERROR:s	Note
ErrorContentFixed	Controls the contents for ERROR:s. Fixed font.	ErrorContent
FatalBanner	Controls the banner for FATAL:s	Note
FatalContent	Controls the contents of FATAL:s	Note
FatalContentFixed	Controls the contents for FATAL:s. Fixed font.	FatalContent

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.9 Style Elements Affecting Tables and Batch Output

Style Element	Description	Inherits from
Tables and Batch Output		
Output	Controls basic output forms. This is generally used to control the borders (using the FRAME=, RULES=, and individual border control attributes), cell spacing, cell padding, and background color.	Container *
Table	Controls overall table style	Output
Batch	Controls batch mode output	Output
TableHeaderContainer	Places and controls the box around all column headings (RTF only)	Container *
		
TableFooterContainer	Places and controls the box around all column footers (RTF only)	Container *

Style Element	Description	Inherits from
Tables and Batch Output		
		
ColumnGroup	Places and controls the box around groups of columns (RTF only)	Container *
		

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.10 Style Elements Affecting Data Cells in Tables

Style Element	Description	Inherits from
Table Data Cells		
Cell	Controls data, header, and footer cells	Container *
Data	Default style for data cells	Cell
DataFixed	Default style for data cells that request a fixed font	Data
DataEmpty	Controls emphasized data cells	Data
DataEmphasis	Controls emphasized data cells	Data
DataEmphasisFixed	Controls emphasized data cells that request a fixed font	DataEmphasis

Style Element	Description	Inherits from
Table Data Cells		
DataStrong	Controls strong (more emphasized) data cells	Data
DataStrongFixed	Controls strong (more emphasized) data cells that request a fixed font	DataStrong

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.11 Style Elements Affecting Header and Footer Cells

Style Element	Description	Inherits from
Table Header and Footer Cells		
HeadersAndFooters	Controls table headers and footers	Cell *
Header	Controls the headers of a table	HeadersAndFooters
HeaderFixed	Controls the header of a table that request a fixed font	Header
HeaderEmpty	Controls empty table header cells	Header
HeaderEmphasis	Controls emphasized table header cells that request a fixed font	Header
HeaderEmphasisFixed	Controls emphasized table header cells that request a fixed font	HeaderEmphasis
HeaderStrong	Controls strong (more emphasized) table header cells	Header
HeaderStrongFixed	Controls strong (more emphasized) table header cells	HeaderStrong
RowHeader	Controls row headers	Header
RowHeaderFixed	Controls row headers that request a fixed font	RowHeader
RowHeaderEmpty	Controls empty row headers	RowHeader
RowHeaderEmphasis	Controls emphasized row headers	RowHeader
RowHeaderEmphasisFixed	Controls emphasized row headers that request a fixed font	RowHeaderEmphasis
RowHeaderStrong	Controls strong (more emphasized) row headers	RowHeader
RowHeaderStrongFixed	Controls strong (more emphasized) row headers that request a fixed font	RowHeaderStrong
Footer	Controls table footers	HeadersAndFooters
FooterFixed	Controls table footers that request a fixed font	Footer

Style Element	Description	Inherits from
Table Header and Footer Cells		
FooterEmpty	Controls empty table footers	Footer
FooterEmphasis	Controls emphasized table footers	Footer
FooterEmphasisFixed	Controls emphasized table footers that request a fixed font	FooterEmphasis
FooterStrong	Controls strong (more emphasized) table footers	Footer
FooterStrongFixed	Controls strong (more emphasized) table footers that request a fixed font	FooterStrong
RowFooter	Controls a row footer (label)	Footer
RowFooterFixed	Controls a row footer (label) that request a fixed font	RowFooter
RowFooterEmpty	Controls an empty row footer (label)	RowFooter
RowFooterEmphasis	Controls an emphasized row footer (label)	RowFooter
RowFooterEmphasisFixed	Controls an emphasized row footer (label) that request a fixed font	RowFooterEmphasis
RowFooterStrong	Controls a strong (more emphasized) row footer (label)	RowFooter
RowFooterStrongFixed	Controls a strong (more emphasized) row footer (label) that requests a fixed font	RowFooterStrong

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table A4.12 Style Elements Affecting PROC TABULATE Captions

Style Element	Description	Inherits from
PROC TABULATE Captions		
Caption	Controls captions in PROC TABULATE	HeadersAndFooters *
BeforeCaption	Caption that comes before a table	Caption
AfterCaption	Caption that comes after a table	Caption

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Style Elements Affecting Template-Based Graphics

The following style elements affect template-based graphics and can be specified by Graph Template Language appearance options or used in styles. Template-based graphics include all SAS/GRAPH output where a compiled ODS template of type STATGRAPH is used to produce graphical output. Supplied templates are stored in SASHELP.TMPLMST. Device drivers and some global statements such as SYMBOL, PATTERN, AXIS, and LEGEND have no affect on this form of graphics. Common SAS/GRAPH procedures that produce template-based graphics are SGPLOT, SGPANEL, and SGRENDER in addition to many SAS/STAT, SAS/ETS, and SAS/QC procedures. ODS graphics always produce output as image files and use the ODS GRAPHICS statement to control the graphical environment.

Certain style elements were created to be used with specific plots or graphs. For example, the style element GraphFit2 is best used to modify secondary fit lines. The style element GraphConfidence2 was created to modify secondary confidence bands. The table below lists each style element, the portion of the graph it affects or was created to use with, and the default attribute values. Attribute values can be changed with PROC TEMPLATE, as stated above.

For complete documentation on the style attributes that can be specified in each style element, see "Style Attributes Overview" on page 498.

Table A4.13 Graph Style Elements: General Graph Appearance

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
Graph	Graph size and outer border appearance	OutputWidth	Not set
		OutputHeight	Not set
		BorderColor	Inherited
		BorderWidth	Inherited
		CellPadding	0
		CellSpacing	Inherited
GraphAnnoLine	Annotation lines	ContrastColor	GraphColors("gcdata")
		LineStyle	1
		LineThickness	1px
GraphAnnoShape	Annotation closed shapes such as circles, and squares	Color	GraphColors("gcdata"
		ContrastColor	GraphColors("gcdata")
		LineThickness	2px
		LineStyle	1
		Transparency	Not set
GraphAnnoText	Annotation text	Font	GraphFonts("annofont")
		or *font-attributes**	Not set
		Color	GraphColors("gtext")
GraphAxisLines	X, Y and Z axis lines	ContrastColor	GraphColors("gaxis")
		LineStyle	1
		LineThickness	1px
		TickDisplay	"Outside"

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphBackground	Background of the graph	Color Transparency	Colors("docbg") Not set
GraphBorderLines	Border around graph wall, legend border, borders to complete axis frame	ContrastColor LineThickness LineStyle	GraphColors("gborderlines") 1px 1
GraphDataText	Text font and color for point and line labels	Font or *font-attributes** Color	GraphFonts("GraphDataFont") Not set GraphColors("gtext")
GraphFootnoteText	Text font and color for footnote(s)	Font or *font-attributes** Color	GraphFonts("GraphFootnoteFont") Not set GraphColors("gtext")
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	ContrastColor LineStyle LineThickness Transparency DisplayOpts	GraphColors("ggrid") 1 1px Not set "Auto"
GraphHeaderBackground	Background color of the legend title	Color Transparency	Colors("gheader") Not set
GraphLabelText	Text font and color for axis labels and legend titles	Font or *font-attributes** Color	GraphFonts("GraphLabelFont") Not set GraphColors("glabel")
GraphLegendBackground	Background color of the legend	Color Transparency	Colors("glegend") Not set
GraphOutlines	Outline properties for fill areas such as bars, pie slices, box plots, ellipses, and histograms	Color ContrastColor LineStyle LineThickness	GraphColors("goutlines") GraphColors("goutlines") 1 1px
GraphReference	Horizontal and vertical reference lines and drop lines	ContrastColor LineStyle LineThickness	GraphColors("greferencelines") 5 1px
GraphTitleText	Text font and color for title(s)	Font or *font-attributes** Color	GraphFonts("GraphTitleFont") Not set GraphColors("gtext")
GraphUnicodeText	Text font for unicode values	Font or*font-attributes** Color	GraphFont("GraphUnicodeFont") Not set GraphColors("gtext")

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphValueText	Text font and color for axis tick values and legend values	Font	GraphFonts("GraphValueFont")
		or *font-attributes**	Not set
		Color	GraphColors("gtext")
GraphWalls	Vertical wall(s) bounded by axes	Color	GraphColors("gwalls")
		Transparency	Not set
		FrameBorder	On
		LineThickness	1px
		LineStyle	1
		ContrastColor	GraphColors("gaxis")

* *Font-attributes* can be one of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.

Table A4.14 Style Elements Affecting Graphical Data Representation

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphBoxMean	Marker for mean	ContrastColor	GraphColors("gcdata")
		MarkerSize	9px
		MarkerSymbol	"Diamond"
GraphBoxMedian	Line for median	ContrastColor	GraphColors("gcdata")
		LineStyle	1
		LineThickness	1px
GraphBoxWhisker	Box whiskers and serifs	ConstrastColor	GraphColors("gcdata")
		LineStyle	1
		LineThickness	1px
GraphConfidence	Primary confidence lines and bands, colors for bands and lines	ContrastColor	GraphColors("gcconfidence")
		Color	GraphColors("gconfidence")
		MarkerSize	7px
		MarkerSymbol	"Diamond"
		LineStyle	2
		LineThickness	1px
GraphConfidence2	Secondary confidence lines and bands, color for bands, amd contrast color for lines	ContrastColor	GraphColors("gcconfidence2")
		Color	GraphColors("gconfidence2")
		MarkerSize	7px
		MarkerSymbol	"Triangle"
		LineStyle	41
		LineThickness	1px
GraphConnectLine	Line for connecting boxes or bars	ContrastColor	GraphColors("connectLine")
		LineStyle	1
		LineThickness	1px

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphDataDefault	Primitives related to non-grouped data items, colors for filled areas, markers, and lines	Color	GraphColors("gdata")
		ContrastColor	GraphColors("gcdata")
		MarkerSymbol	”circle”
		MarkerSize	7px
		LineStyle	1
		LineThickness	1px
		StartColor	GraphColors("gramp3cstart")
		NeutralColor	GraphColors("gramp3cneutral")
		EndColor	GraphColors("gramp3cend")
GraphError	Error line or error bar fill, ContrastColor for lines, Color for bar fill	ContrastColor	GraphColors("gcerror")
		Color	GraphColors("gerror")
		LineStyle	5
		Transparency	Not set
GraphFit	Primary fit lines such as a normal density curve	ContrastColor	GraphColors("gcfit")
		Color	GraphColors("gfit")
		MarkerSize	7px
		MarkerSymbol	”Circle”
		LineStyle	1
		LineThickness	2px
GraphFit2	Secondary fit lines such as a kernel density curve	ContrastColor	GraphColors("gcfit")
		Color	GraphColors("gfit")
		MarkerSize	7px
		MarkerSymbol	“X”
		LineStyle	5
		LineThickness	2px
GraphMissing	Properties for graph items representing missing values	ContrastColor	GraphColors("gcmissing")
		Color	GraphColors("gmissing")
		MarkerSymbol	“square”
		MarkerSize	7px
		LineStyle	1
		LineThickness	1px
		Transparency	Not set
GraphOutlier	Outlier data for the graph	ContrastColor	GraphColors("goutlier")
		Color	GraphColors("gcoutlier")
		MarkerSize	7px
		MarkerSymbol	”Circle”
		LineStyle	42
		LineThickness	2px

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphPrediction	Prediction lines	ContrastColor	GraphColors("gcpredict")
		Color	GraphColors("gpredict")
		LineStyle	4
		LineThickness	2px
		MarkerSize	7px
		MarkerSymbol	"Plus"
GraphPredictionLimits	Fills for prediction limits	ContrastColor	GraphColors("gcpredictlim")
		Color	GraphColors("gpredictlim")
		MarkerSize	7px
		MarkerSymbol	"Chain"
GraphSelection	For interactive graphs, visual properties of selected item. Color for selected fill area, ContrastColor for selected marker or line	ContrastColor	GraphColors("gcdata")
		Color	GraphColors("gdata")
		MarkerSymbol	“Square”
		MarkerSize	11px
		LineStyle	1
		LineThickness	5px
ThreeColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor	GraphColors("gconramp3start")
		NeutralColor	GraphColors("gconramp3cneutral")
		EndColor	GraphColors("gconramp3end")
ThreeColorRamp	Gradient contours, surfaces, markers, and data labels with continuous color response	StartColor	GraphColors("gramp3cstart")
		NeutralColor	GraphColors("gramp3cneutral")
		EndColor	GraphColors("gramp3cend")
TwoColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor	GraphColors("gconramp2cstart")
		EndColor	GraphColors("gconramp2cend")
TwoColorRamp	Gradient contours, surfaces, markers, and data labels with continuous color response	StartColor	GraphColors("gramp2cstart")
		EndColor	GraphColors("gramp2cend")

Table A4.15 Graphical Style Elements: Data Related (Grouped)

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphData1	Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines.	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata1") GraphColors("gcdata1") "Circle" 1
GraphData2	Primitives related to 2nd grouped data items	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata2") GraphColors("gcdata2") "Plus" 4
GraphData3	Primitives related to 3rd grouped data items	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata3") GraphColors("gcdata3") "x" 8
GraphData4	Primitives related to 4th grouped data items	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata4") GraphColors("gcdata4") "Triangle" 5
GraphData5	Primitives related to 5th grouped data items	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata5") GraphColors("gcdata5") "Square" 14
GraphData6	Primitives related to 6th grouped data items	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata6") GraphColors("gcdata6") "Asterisk" 26
GraphData7	Primitives related to 7th grouped data items	Color ContrastColor MarkerSymbol LineStyle	GraphColors("gdata7") GraphColors("gcdata7") "Diamond" 15
GraphData8	Primitives related to 8th grouped data items	Color ContrastColor LineStyle	GraphColors("gdata8") GraphColors("gcdata8") 20
GraphData9	Primitives related to 9th grouped data items	Color ContrastColor LineStyle	GraphColors("gdata9") GraphColors("gcdata9") 41
GraphData10	Primitives related to 10th grouped data items	Color ContrastColor LineStyle	GraphColors("gdata10") GraphColors("gcdata10") 42

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphData11	Primitives related to 11th grouped data items	Color ContrastColor LineStyle	GraphColors("gdata11") GraphColors("gcdata11") 2
GraphData12	Primitives related to 12th grouped data items	Color ContrastColor	GraphColors("gdata12") GraphColors("gcdata12")

Table A4.16 Display Style Elements

Style Element	Portion of Graph Affected	Recognized Attributes	Possible Values
GraphAltBlock	Alternate fill color for block plots	Color	GraphColors("gablock")
GraphBand	Display options for confidence bands	DisplayOpts	"Fill "
GraphBox	Display options for box plots	DisplayOpts CapStyle Connect	"Fill caps mean Median outliers " "Serif" "Mean"
GraphBlock	Fill color for block plots	Color	GraphColors("gblock")
GraphEllipse	Display options for confidence ellipses	DisplayOpts	"Outline"
GraphHistogram	Display options for histograms	DisplayOpts	"Fill outline"

Style Elements Affecting Device-Based Graphics

Device-based graphics are all SAS/GRAPH output where there is a user-specified or default device (DEVICE= option) that controls certain aspects of the graphical output. Supplied device drivers are stored in the SASHELP.DEVICES catalog. Examples of devices drivers are SASPRTC, GIF, WIN, ACTIVEX, PDF, and SVG. Common SAS/GRAPH procedures that produce device-based graphics are GPLOT, GCHART, and GMAP. Most device-based graphics produce produce a GRSEG catalog entry as output and use the GOPTIONS statement to control the graphical environment.

For complete documentation on the style attributes that can be specified in each style element, see "Style Attributes Overview" on page 498.

Note: These style elements affect device-based graphics only when the GSTYLE system option is in effect (this is is the default for SAS 9.2). If the NOGSTYLE system option is specified, graphs do not use any style information. For more information about the GSTYLE system option, see *SAS Language Reference: Dictionary*. △

Table A4.17 Device-Based Graph Style Elements: General Graph Appearance

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
DropShadowStyle	Used with text types	Color	GraphColors("gshadow")
Graph	Graph size and outer border appearance	OutputWidth	Not set
		OutputHeight	Not set
		BorderColor	Inherited
		BorderWidth	Inherited
		CellPadding	0
		CellSpacing	Inherited
GraphAxisLines	X, Y, and Z axis lines	Color	GraphColors("gaxis")
		LineStyle	1
		LineThickness	1px
GraphBackground	Background of the graph	Transparency	Not set
		BackgroundColor	Colors("docbg")
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set
		VerticalAlign	Not set
		TextAlign	Not set
GraphBorderLines	Border around graph wall, legend border, borders to complete axis frame	Color	GraphColors("gborderlines")
		LineThickness	1px
		LineStyle	1
GraphCharts	All charts within the graph	Transparency	Not set
		BackgroundColor	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set
		VerticalAlign	Not set
		TextAlign	Not set
GraphDataText	Text font and color for point and line labels	Font or *font-attributes**	GraphFonts("GraphDataFont")
		Color	Not set
			GraphColors("gtext")

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphFloor	3D floor	BackgroundColor	GraphColors("gfloor")
		Transparency	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set
		VerticalAlign	Not set
		TextAlign	Not set
GraphFootnoteText	Text font and color for footnotes	Font or *font-attributes**	GraphFonts("GraphFootnoteFont")
		Color	Not set
			GraphColors("gtext")
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	Color	GraphColors("ggrid")
		LineStyle	1
		LineThickness	1px
		Transparency	.5
		displayopts	"Auto"
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	Color	GraphColors("ggrid")
		LineStyle	1
		LineThickness	1px
		Transparency	.5
		displayopts	"Auto"
GraphLegendBackground	Background color of the legend	Color	Colors("glegend")
		Transparency	Not set
GraphOutlines	Outline properties for fill areas such as bars, pie slices, and box plots.	Color	GraphColors("goutlines")
		LineStyle	1
		LineThickness	1px
GraphTitleText	Text font and color for titles	Font or *font-attributes**	GraphFonts("GraphTitleFont")
		Color	Not set
			GraphColors("gtext")

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphValueText	Text font and color for axis tick values and legend values	Font or *font-attributes**	GraphFonts("GraphValueFont")
		Color	Not set
			GraphColors("gtext")
GraphWalls	Vertical walls bounded by axes	Transparency	Not set
		BackgroundColor	GraphColors("gwalls")
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set

* *Font-attributes* can be one of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.

Table A4.18 Style Elements Affecting Device-Based Non-Grouped Graphical Data Representation

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
ThreeColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor	GraphColors("gconramp3start")
		NeutralColor	GraphColors("gconramp3cneutral")
		EndColor	GraphColors("gconramp3end")
ThreeColorRamp	Gradient contours, surfaces, markers, nad data labels with continuous color response	StartColor	GraphColors("gramp3cstart")
		NeutralColor	GraphColors("gramp3cneutral")
		EndColor	GraphColors("gramp3cend")
TwoColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor	GraphColors("gconramp2cstart")
		EndColor	GraphColors("gconramp2cend")
TwoColorRamp	Gradient contours, surfaces, markers, and data labels with continuous color response	StartColor	GraphColors("gramp2cstart")
		EndColor	GraphColors("gramp2cend")

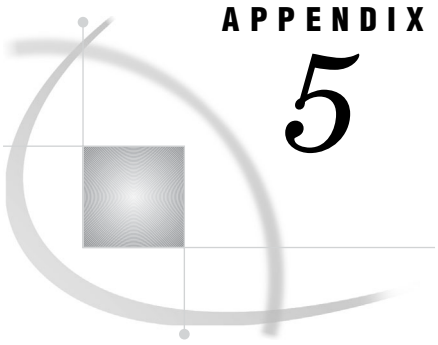
Table A4.19 Style Elements Affecting Device-Based Grouped Graphical Data Representation

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData1	Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines.	Color	GraphColors("gdata1")
		ContrastColor	GraphColors("gcdata1")
		MarkerSymbol	"Circle"
		LineStyle	1
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData2	Primitives related to 2nd grouped data items	Color	GraphColors("gdata2")
		ContrastColor	GraphColors("gcdata2")
		MarkerSymbol	"Plus"
		LineStyle	4
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData3	Primitives related to 3rd grouped data items	Color	GraphColors("gdata3")
		ContrastColor	GraphColors("gcdata3")
		MarkerSymbol	"X"
		LineStyle	8
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData4	Primitives related to 4th grouped data items	Color	GraphColors("gdata4")
		ContrastColor	GraphColors("gcdata4")
		MarkerSymbol	"triangle"
		LineStyle	5
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData5	Primitives related to 5th grouped data items	Color	GraphColors("gdata5")
		ContrastColor	GraphColors("gcdata5")
		MarkerSymbol	"square"
		LineStyle	14
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData6	Primitives related to 6th grouped data items	Color	GraphColors("gdata6")
		ContrastColor	GraphColors("gcdata6")
		MarkerSymbol	"Asterisk"
		LineStyle	26
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData7	Primitives related to 7th grouped data items	Color	GraphColors("gdata7")
		ContrastColor	GraphColors("gcdata7")
		MarkerSymbol	"Diamond"
		LineStyle	15
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData8	Primitives related to 8th grouped data items	Color	GraphColors("gdata8")
		ContrastColor	GraphColors("gcdata8")
		MarkerSymbol	Not set
		LineStyle	20
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData9	Primitives related to 9th grouped data items	Color	GraphColors("gdata9")
		ContrastColor	GraphColors("gcdata9")
		MarkerSymbol	Not set
		LineStyle	41
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData10	Primitives related to 10th grouped data items	Color	GraphColors("gdata10")
		ContrastColor	GraphColors("gcdata10")
		MarkerSymbol	Not set
		LineStyle	42
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData11	Primitives related to 11th grouped data items	Color	GraphColors("gdata11")
		ContrastColor	GraphColors("gcdata11")
		MarkerSymbol	Not set
		LineStyle	2
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData12	Primitives related to 12th grouped data items	Color	GraphColors("gdata12")
		ContrastColor	GraphColors("gcdata12")
		MarkerSymbol	Not set
		LineStyle	Not set
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set



APPENDIX
5

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *Base SAS Procedures Guide*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *Step-by-Step Programming with Base SAS Software*

The recommended reading list from *SAS Press* includes:

- *The Little SAS Book: A Primer, Revised Second Edition*
- *Output Delivery System: The Basics*

For a complete list of SAS publications, go to **support.sas.com/bookstore**. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative at:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: 1-800-727-3228
Fax: 1-919-531-9439
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/bookstore**

Customers outside the United States and Canada, please contact your local SAS office for assistance.

Glossary

access mode
the level of access that a user has to an item store. The possible access modes are read, write, and update. See also item store.

ActiveX
an image format that stores interactive graphical output that was generated using an ActiveX control. The output is stored in a single file. You can right-click on any graphical output that is generated in this format and change multiple options. Any options that you change are reflected only in the output, not in the task window selections that you made to set up the chart. In order to use the ActiveX format, you will need to install the ActiveX control.

ActiveX control
a type of Web application that is developed specifically for the Windows operating environment. ActiveX controls can provide Web users with interactive capabilities.

after-note
in ODS, a note that is displayed after an output object each time the output object is displayed. The text is assigned to an output object by the procedure that produced the object. See also output object, ODS document, and before-note.

aggregate storage location
a location on an operating system that can contain a group of distinct files. On different operating systems, different terms (such as directory, folder, or partitioned data set) are used to refer to an aggregate storage location.

aliasing
a visual effect in computer-generated images that produces several types of rendering problems, such as jagged edges along straight lines or polygon boundaries. Aliasing can occur when you try to render an object smaller than pixel size or a very narrow object. In a complex scene, fine details are sometimes lost or distorted beyond recognition due to aliasing. See also anti-aliasing.

annotation
a label, marker, or note that is not obtained from the data but is placed on a graph independently. Such annotations might or might not be linked to data values in the plot.

anti-aliasing
a technique that tries to overcome the problems of aliasing, especially when rendering three-dimensional graphics. Aliasing problems include jagged edges, incorrect rendering of thin or small objects, and confused rendering of complex objects. See also aliasing.

before-note
in ODS, a note that is displayed before an output object each time the output object is displayed. The text is assigned to the output object by the procedure that produced the object. See also after-note, output object, and ODS document.

cell
in general, the intersection of a row and a column in a table. In some SAS procedures such as PROC TABULATE and PROC FREQ, the value of each cell is a summary statistic for the input data set. The contents of the cell are described by the page, row, and column that contain the cell.

cellvalue
one of the possible values that PROC FREQ can produce for a crosstabulation table. Cellvalues are defined by the DEFINE CELLVALUE statement in a crosstabulation table template.

column attribute
a formatting property that controls aspects of a column, such as the appearance of the cells contents, presentation of data panels, and customization of column headers. Column attributes have a reserved name and value defined in ODS.

crosstabulation table
a frequency table that shows combined frequency distributions or other descriptive statistics for two or more variables. See also frequency table.

data component
a form, similar to a SAS data set, that contains the results (numbers and characters) of a DATA step or PROC step that supports ODS.

device-based graphic
a graph created with SAS/GRAPH software for which a user-specified or default device (DEVICE= option) controls certain aspects of the graphical output.

dictionary variable
a type of memory variable that consists of an array that contains a list of numbers or text strings that can be identified by a key. A dictionary variable has, as part of it's name, a preceding '$' symbol and a subscript that contains a text string. The text string within the subscript is called a key. For example, the following dictionary variable identifies the entry in the $MyDictionary variable that contains the text-string 'dog': $MyDictionary['dog']. See also ODS event, list variable, memory variable, and scalar variable.

DOCUMENT destination
an ODS destination that produces a hierarchy of output objects. The DOCUMENT destination enables you to render multiple ODS output formats without rerunning a PROC step or DATA step, and it gives you more control over the structure of the output. See also ODS destination.

exclusion list
a list that tells ODS which output objects to exclude from a specified ODS destination. See also selection list.

footer attribute
a formatting property that controls aspects of a footer, such as the appearance of the footer contents and the placement of the footer. The footer attribute has a reserved name and value definted in ODS. See also header attribute.

frequency table
a table that lists each of the distinct values that a variable has within all of the observations in a SAS data set. For each value, the table also lists the number of observations in which the variable has that value.

graph
a visualization created by SAS software. A graph that is created by the ODS Graphics system can contain titles, footnotes, legends, and one or more cells, and is typically saved as an image or an SGE file.

graph segment
in ODS, a file type or output object that contains a graph. Graphs are created in some SAS procedures, including those in SAS/GRAPH. The graph output object is referenced as a GRSEG. See also output object.

Graph Template Language
an extension to the Output Delivery System (ODS) that enables users to create sophisticated analytical graphics. Short form: GTL.

GTL
See Graph Template Language.

header attribute
a formatting property that controls aspects of a header, such as the appearance of the header contents and the placement of the header. The header attribute has a reserved name and value defined in ODS. See also footer attribute.

HTML
See HyperText Markup Language.

HyperText Markup Language
a coding system in which the codes indicate the layout and style of the text in a text file. Other HTML codes enable you to embed electronic objects such as images, sounds, video streams, and applets (small software applications) into HTML documents. All Web browsers can process HTML documents. Short form: HTML.

inline formatting
a feature of the Output Delivery System (ODS) that allows you to insert simple formatting text into ODS output by using the ODS ESCAPECHAR statement.

item store
a SAS data set that consists of pieces of information that can be accessed independently. The contents of an item store are organized in a directory tree structure, which is similar to the directory structures that are used by UNIX System Services or by Windows. For example, a particular value might be stored and located using a directory path (root_dir/sub_dir/value). The SAS Registry is an example of an item store. See also template store.

legend
a visual key to graphic elements in a graph. The legend consists of the legend value, the legend value description, the legend label, and the legend frame.

list variable
a type of memory variable that consists of an array that contains a list of numbers or text strings that are indexed. A list variable has, as part of its name, a preceding '$' symbol and a subscript that is empty or contains a number or numeric variable. The

number within the subscript is called an index. For example, the list variable $Mylist[2] identifies the second entry in the list variable $Mylist. In this case, the index is 2. See also dictionary variable, ODS event, memory variable, and scalar variable.

LISTING destination
an ODS destination that produces traditional SAS output (monospace format). See also listing output and ODS destination.

marker
(1) a symbol such as a circle, triangle, or diamond that is used to indicate the location of a data point in a plot. (2) a type of annotation that is used in SAS/GRAPH ODS Graphics Editor to highlight particular data in a plot or graph.

markup family
See ODS markup family.

markup language
a set of codes that are embedded in text in order to define layout and certain content.

measured RTF
a tagset that enables users to specify how and where page breaks occur in RTF documents and when to place titles and footnotes into the body of a page.

memory variable
within an ODS event, an area of memory that contains numeric data, character data, or lists of numeric or character data. A memory variable can be classified as a dictionary variable if it is created with a subscript that contains a key, or a list variable if it is created with a subscript that is empty or contains an index. If you do not specify a key or an index, then the memory variable is a numeric or character scalar variable, depending on the variable's value. See also dictionary variable, list variable, scalar variable, and ODS event.

ODS
See Output Delivery System.

ODS destination
a designation that the Output Delivery System uses to generate a specific type of output. Types of ODS destinations include but are not limited to HTML, XML, listing, Postscript, RTF, and SAS data sets.

ODS document
a hierarchy of output objects created by the DOCUMENT procedure. These objects are in an unformatted form and are placed in a SAS item store. See also item store and output object.

ODS document path
the location of an entry within an ODS document. See also ODS document and ODS entry.

ODS entry
an item in an ODS document. An ODS entry can be either a link, an output object, a file, or a partitioned data set.

ODS event
within a tagset definition, an action that causes output to be generated. Events are usually triggered by SAS but can also be triggered by other events.

ODS Graphics system
an extension to ODS that is used to create analytical graphs using the Graph Template Language.

ODS Graphics template
a template for graphics that is created by the TEMPLATE procedure and that contains the definition of a graph.

ODS markup family
a group of ODS statements that produce SAS output that is formatted using a markup language such as HTML (HyperText Markup Language), XML (Extensible Markup Language), and LaTeX. SAS supplies many markup languages for you to use, ranging from DOCBOOK to TROFF. You can specify a markup language that SAS supplies, or you can create one of your own and store it as a user-defined markup language. See also ODS destination and ODS printer family.

ODS output
formatted output that is generated by any of the ODS destinations. For example, the OUTPUT destination produces SAS data sets, the LISTING destination produces listing output, and the HTML destination produces output that is formatted in Hypertext Markup Language.

ODS package
a container for information or digital content that is generated or collected for delivery to a consumer. ODS packages allow ODS destinations to use the SAS Publishing Framework.

ODS printer family
a group of ODS statements that produce output in a format such as PostScript (PS), PDF, or PCL that is suitable for printing on a high-resolution printer.

ODS style
See style definition.

Output Delivery System
a component of SAS software that can produce output in a variety of formats such as markup languages (HTML, XML), PDF, listing, RTF, Postscript, and SAS data sets. Short form: ODS.

output object
a programming object that contains the data that is generated by a DATA step or a PROC step and which can also contain a table definition that provides information about how to format that data.

printer family
See ODS printer family.

Publishing Framework
a component of SAS Integration Technologies that enables both users and applications to publish SAS files (including data sets, catalogs, and database views), other digital content, and system- generated events to a variety of destinations. The Publishing Framework also provides tools that enable both users and applications to receive and process published information.

replay
in ODS, the regeneration of output by the DOCUMENT procedure, in the same or different format, without rerunning analyses or data queries.

root file location
the top level of a file location in an ODS document. A root file location is not contained within another file location and does not have a name assigned to it. A root file location is similar to the root directory of a Windows operating environment.

SASEDOC engine
a SAS engine that associates a SAS libref (library reference) with one or more ODS output objects that are stored in an ODS document.

scalar variable
a type of memory variable that contains one-dimensional numeric or character data. Once created, scalar variables are globally available in all events. See also dictionary variable, list variable, and memory variable.

selection list
a list that tells ODS which output objects to send to a specified ODS destination. See also exclusion list.

stream variable
within an ODS event, a temporary item store that contains output. While the stream variable is open, all output is directed to it until it is closed. See also item store, ODS event, and tagset.

style
See style definition.

style attribute
a visual property, such as color, font properties, and line characteristics, that has a reserved name and value defined in ODS. Style attributes are collectively referenced by a style element within a style definition. See also style definition, style element, and ODS event.

style definition
a template that specifies instructions for the presentation aspects (color, font face, font size, and so on) of your SAS output. This template determines the overall appearance of the documents that use it. Each style definition is composed of style elements. Style definitions have no effect on the LISTING destination, which produces plain text output. See also style element and table definition.

style definition inheritance
the concept that a child style definition receives all the style elements, attributes, and statements that are specified in its parent style definition unless the child style definition overrides them.

style element
a collection of style attributes that each pertain to a particular part of some ODS output. For example, a style element might contain instructions for the presentation of individual table cells. Within the style element, each style attribute specifies a value for one aspect of the presentation. For example, the FLYOVER= attribute specifies the text to display in a tool tip for a cell, and the PRETEXT= attribute specifies the text to place before a cell. See also style definition and table element.

style element inheritance
the concept that a child style element receives all of the style attributes that are specified in its parent style element, unless the child style element overrides those attributes.

table attribute
a formatting property such as layout of headers, line spacing, and layout of rows and columns, that has a reserved name and value defined in ODS. See also table definition and table element.

table definition
a set of instructions that describe how to format output in the Output Delivery System (ODS).

table element
a collection of table attributes that each pertain to a particular column, header, or footer in a table in ODS output.

table template
a template that describes how to display the output for a tabular output object. A table template determines the order of table headers and footers, the order of columns, and the overall appearance of the output object that uses it. Each table template contains or references table elements.

tagset
a template that defines how to create a type of markup language output from a SAS format. Tagsets produce markup output such as Hypertext Markup Language (HTML), Extensible Markup Language (XML), and LaTeX. See also markup language.

tagset definition
a template that specifies instructions for creating a markup language for your SAS output. The resulting output contains embedded instructions in order to define layout and some content. Each tagset definition contains event definitions and event attributes that control the generation of the output. SAS provides tagset definitions for a variety of markup languages. You can use the TEMPLATE procedure to modify any of these SAS tagsets or to create your own tagsets. See also ODS markup family.

template
a description of how output should appear when it is formatted.

template store
an item store that contains definitions that were created by the TEMPLATE procedure. Definitions that SAS provides are in the item store Sashelp.Tmplmst. You can store definitions that you create in any template store to which you have write access. See also item store.

Index

C

D

E

F

G

H

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.

SAS® Publishing Delivers!

Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at support.sas.com/bookstore.

SAS® Press

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

support.sas.com/saspress

SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – **free** on the Web.
- Hard-copy books.

support.sas.com/publishing

SAS® Publishing News

Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site.

support.sas.com/spn

sas | THE POWER TO KNOW®