# Contents

## PART 1  Introduction  1

**Chapter 1 / Introduction**  
Overview of the Output Delivery System  3  
Overview of How ODS Works  4

## PART 2  Customizing Procedure Output  9

**Chapter 2 / Working with General Procedure Output**  
Customized Output for an Output Object  11  
Customizing Titles and Footnotes  12  

**Chapter 3 / Using Selection and Exclusion Lists**  
Overview  13  
Dictionary  14  

**Chapter 4 / Other SAS Statements That Are Used with ODS Procedures**  
Dictionary  37

## PART 3  The DOCUMENT Procedure  87

**Chapter 5 / ODS DOCUMENT Statement**  
Dictionary  89  

**Chapter 6 / DOCUMENT Procedure**  
Overview: DOCUMENT Procedure  94  
Concepts: DOCUMENT Procedure  95  
Syntax: DOCUMENT Procedure  98  
Usage: DOCUMENT Procedure  148  
Examples: DOCUMENT Procedure  162

## PART 4  The ODSLIST Procedure  191

**Chapter 7 / ODSLIST Procedure**  
Overview: ODSLIST Procedure  193  
Syntax: ODSLIST Procedure  194
## Contents

**PART 5**  The ODSTABLE Procedure  229

*Chapter 8 / ODSTABLE Procedure*  231
- Overview: ODSTABLE Procedure  231
- Concepts: ODSTABLE Procedure  232
- Syntax: ODSTABLE Procedure  233
- Usage: ODSTABLE Procedure  264
- Examples: ODSTABLE Procedure  268

**PART 6**  The ODSTEXT Procedure  287

*Chapter 9 / ODSTEXT Procedure*  289
- Overview: ODSTEXT Procedure  289
- Syntax: ODSTEXT Procedure  290
- Usage: ODSTEXT Procedure  308
- Examples: ODSTEXT Procedure  311

**PART 7**  The TEMPLATE Procedure  319

*Chapter 10 / Overview*  321
- Introduction to the TEMPLATE Procedure  321
- Terminology: TEMPLATE Procedure  322
- The Backward Compatibility of ODS Templates  323
- Syntax  324
- Using the TEMPLATE Procedure  325
- PROC TEMPLATE Statements by Category  330
- Where to Go from Here  332

*Chapter 11 / TEMPLATE Procedure: Managing Template Stores*  333
- Overview: TEMPLATE Procedure: Managing Template Stores  333
- Concepts: TEMPLATE Procedure: Managing Template Stores  334
- Syntax: TEMPLATE Procedure: Managing Template Stores  336
- Examples: TEMPLATE Procedure: Managing Template Stores  351

*Chapter 12 / TEMPLATE Procedure: Creating Crosstabulation Table Templates*  359
- Overview: TEMPLATE Procedure: Creating Crosstabulation Table Templates  360
- Concepts: TEMPLATE Procedure: Creating Crosstabulation Table Templates  363
- Syntax: TEMPLATE Procedure: Creating Crosstabulation Table Templates  364
- Usage: TEMPLATE Procedure: Creating Crosstabulation Table Templates  387
- Examples: TEMPLATE Procedure: Creating Crosstabulation Table Templates  390

*Chapter 13 / TEMPLATE Procedure: Creating ODS Graphics*  419
- Introduction to the Graph Template Language  419
## Contents

### Chapter 14 / TEMPLATE Procedure: Creating a Style Template
- Overview: TEMPLATE Procedure: Creating a Style Template .......................................................... 425
- Concepts: TEMPLATE Procedure: Creating a Style Template ....................................................... 426
- Syntax: TEMPLATE Procedure: Creating a Style Template ............................................................ 428
- Usage: TEMPLATE Procedure: Creating a Style Template ............................................................. 446
- Examples: TEMPLATE Procedure: Creating a Style Template ....................................................... 459

### Chapter 15 / TEMPLATE Procedure: Creating Tabular Templates
- Overview: TEMPLATE Procedure: Creating Tabular Templates .................................................... 515
- Concepts: TEMPLATE Procedure: Creating Tabular Templates .................................................... 516
- Syntax: TEMPLATE Procedure: Creating Tabular Templates .......................................................... 520
- Usage: TEMPLATE Procedure: Creating Tabular Templates .......................................................... 548
- Examples: TEMPLATE Procedure: Creating Tabular Templates .................................................... 551

### Chapter 16 / Tabular Attributes
- Understanding Table Templates, Table Elements, and Table Attributes ...................................... 591
- Column Attributes ......................................................................................................................... 592
- Header and Footer Attributes ....................................................................................................... 608
- Table Attributes ............................................................................................................................ 620

### Chapter 17 / TEMPLATE Procedure: Creating Markup Language Tagsets
- Overview: TEMPLATE Procedure: Creating Markup Language Tagsets ........................................ 633
- Concepts: TEMPLATE Procedure: Creating Markup Language Tagsets ...................................... 634
- Syntax: TEMPLATE Procedure: Creating Markup Language Tagsets .......................................... 642
- Usage: TEMPLATE Procedure: Creating Markup Language Tagsets .......................................... 683
- Examples: TEMPLATE Procedure: Creating Markup Language Tagsets ...................................... 692

### PART 8 ODS Styles Reference

#### Chapter 18 / Overview
- Understanding Styles, Style Elements, and Style Attributes ....................................................... 719
- Using Styles with Base SAS Procedures ......................................................................................... 725

#### Chapter 19 / Style Templates
- Viewing ODS Styles Supplied by SAS ............................................................................................ 727
- Program for Viewing Multiple Styles ............................................................................................. 729
- Table of Suggested ODS Styles ..................................................................................................... 730
- ODS Styles Gallery ......................................................................................................................... 733

#### Chapter 20 / Style Elements
- ODS Style Elements ....................................................................................................................... 797
- Style Elements Affecting Template-Based Graphics ...................................................................... 808
- Style Elements Affecting Device-Based Graphics ......................................................................... 817

#### Chapter 21 / Style Attributes
- Overview ......................................................................................................................................... 827
- Style Attributes Tables .................................................................................................................. 828
- Style Attributes Detailed Information ............................................................................................ 855
- Style Attributes Values .................................................................................................................. 890
Appendix 1 / Output Object Table Names

- ODS Table Names Produced by Base SAS Procedures: 899
- ODS Table Names Produced by Base SAS High-Performance Procedures: 914
- ODS Table Names Produced by Base SAS Statistical Procedures: 914
- ODS Table Names Produced by SAS/STAT Procedures: 915
- ODS Table Names Produced by SAS/STAT High-Performance Procedures: 919
- ODS Table Names Produced by SAS Enterprise Miner High-Performance Procedures: 920
- ODS Table Names Produced by SAS/ETS Procedures: 921
- ODS Table Names Produced by SAS/ETS High-Performance Procedures: 923
- ODS Table Names Produced by SAS/QC Procedures: 923
- ODS Table Names Produced by SAS/OR Procedures: 924
- ODS Table Names Produced by Graph Algorithms and Network Analysis Procedure: 925

Appendix 2 / Lua License

- Lua License: 927
PART 1

Introduction

Chapter 1

Introduction 3
Introduction

Overview of the Output Delivery System

The Output Delivery System (ODS) gives you great flexibility in generating, storing, and reproducing SAS procedures and DATA step output, with a wide range of formatting options.

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 style for all of your output, or you can customize a single table or graph.

You can use ODS to accomplish the following tasks:

Create reports for viewers or browsers.
   With ODS, you can use ODS destination statements to create output in popular formats such as HTML, PDF, and RTF. For example, you can use the ODS PDF statement to create PDF files for viewing with Adobe Acrobat or for printing. You can use the ODS EPUB statement to create output for e-book readers. The ODS RTF statement creates output for Microsoft Word. For complete documentation on the ODS destination statements, see “Dictionary of ODS Language Statements” in SAS Output Delivery System: User’s Guide.

Customize the report contents.
   ODS enables you to modify the contents of your output. With ODS, you can embed graphics, select specific cell contents to display, and create embedded links in tables and graphs. You can select specific tables or graphs from procedure output to be printed or you can exclude them. You can create SAS data sets directly from tables or graphics.

Customize the presentation.
   ODS enables you to change the appearance of your output. You can change the colors, fonts, and borders of your output. You can customize the layout, format, headers, and style. You can add images and embedded URLs.
Create more accessible SAS output.

The ODS EPUB and ODS EPUB3 destinations are the recommended destinations for creating SAS output that is accessible to the broadest audience. They create e-books that utilize many of the accessibility features of the EPUB specification. These features enable e-book readers such as iBooks to present e-books so that they adapt to the needs of users with a wide range of abilities. For example, when reading an e-book created by the ODS EPUB and ODS EPUB3 destinations using iBooks on an iPad, users can adjust font size, color schemes, and magnification. They can also access the text using assistive technologies such as the Voiceover screen reader and refreshable braille displays.

Overview of How ODS Works

Components of SAS Output

The PROC or DATA step supplies data and the name of the 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. The following figure shows how SAS produces ODS output.
Overview of How ODS Works

Table 1.1  * List of Tagsets That SAS Supplies and Supports

<table>
<thead>
<tr>
<th>CHTML</th>
<th>CSV</th>
<th>CSVALL</th>
<th>CSVBYLINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>EXCELXP</td>
<td>HTMLPANEL</td>
<td></td>
</tr>
<tr>
<td>MSOFFICE2K</td>
<td>PHTML</td>
<td>TAGSETS.RTF</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2  * Additional Diagnostic Tagsets That SAS Supports

<table>
<thead>
<tr>
<th>EVENT_MAP</th>
<th>NAMEDHTML</th>
<th>SHORT_MAP</th>
<th>STYLE_DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>STYLE_POPUP</td>
<td>TEXT_MAP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Additional tagsets can be found at http://support.sas.com/rnd/base/ods/odsmarkup/.
Features of ODS

ODS delivers output in a variety of formats, and makes the formatted output easy to access.

Important features of ODS include the following:

- ODS combines data with table and graph templates to produce one or more output objects. These objects can be sent to any or all ODS destinations. The currently available ODS destinations can produce the following types of output:
  - 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
  - PowerPoint output that is formatted for use with Microsoft PowerPoint
  - output that is formatted for EPUB e-books

- ODS provides table and graph 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. Most SAS procedures produce multiple output objects. You can easily create HTML output, an output data set, LISTING output, or printer output from any or all 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 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 program, but you do not need to maintain separate programs for each destination. This feature saves you time and system resources by enabling you to produce multiple types of output with a single run of your procedure or data query.

Clearing the Results Window in the SAS Windowing Environment

The ODS HTML destination is turned on by default when you use the SAS windowing environment and SAS Studio. Therefore, the NEWFILE=NONE option is
set by default in the HTML file and all output is sent to a single HTML file. When you clear the Results window manually by clicking on a node in the Results window and selecting **Delete**, or by issuing the `ODSRESULTS;CLEAR` command, nodes are deleted from the Results window. However, the HTML file remains open and the output from subsequent procedure or DATA steps is appended to any previously created HTML output. You can remove the contents of the HTML file, close the HTML file, and reset preferences to the default state by doing one of the following:

- **Using the Command Line**
  1. Select the Results window or the Results Viewer window.
  2. Submit either of these commands in the command line:

     ```
     CLEAR
     ```

     or

     ```
     CLEAR ALL
     ```

- **Using the SAS Windowing Environment**
  1. Select the Results window or the Results Viewer window.
  2. From the Menu, select **Edit ➜ Clear All**.
PART 2

Customizing Procedure Output

Chapter 2  
Working with General Procedure Output .......................... 11

Chapter 3  
Using Selection and Exclusion Lists .............................. 13

Chapter 4  
Other SAS Statements That Are Used with ODS Procedures .... 37
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: ODS” in SAS Output Delivery System: User’s Guide.

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;

A few procedures do not produce ODS output. If you produce a trace record for one of these procedures, no template appears in the trace record. Most procedures use one or more templates to produce their output. Each 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 76.

You can use PROC TEMPLATE to modify templates. When a procedure or DATA step uses a table template, it uses the elements that are defined or referenced in its table template. In most cases, the only way to modify the appearance of your output is by modifying one or more templates, which can include table, graph, style, column, and other templates.
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 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 options that enable you to customize the style of the titles and footnotes. Examples of these style options are BOLD, COLOR=, and FONT=. Because these options control only the presentation of the titles and footnotes, they have no effect on titles or footnotes that go to the LISTING or OUTPUT destination. For a complete list of style options, detailed information about the style options and example code, see the TITLE statement and the FOOTNOTE statement in SAS DATA Step Statements: Reference.

When used with graphics, 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 about this ODS and SAS/GRAPH interaction, see Controlling Titles and Footnotes. For information about adding titles and footnotes to ODS Graphics, see “Adding Titles, Footnotes, and Text Entries to Your Graph” in SAS Graph Template Language: User’s Guide.

For information about titles and footnotes rendered with and without using the graphics option USEGOPT, see “ODS USEGOPT Statement” in SAS Output Delivery System: User’s Guide.
Overview

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.

TIP 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.

You can view the contents of the exclusion and selection lists by using the ODS SHOW statement. The current selection list 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" in SAS Output Delivery System: User’s Guide. For a list of ODS output destinations and explanations of each, see “Understanding ODS Destinations” in SAS Output Delivery System: User’s Guide.

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
written to 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 22, the "ODS EXCLUDE Statement" on page 14, and the "ODS TRACE Statement" on page 76.

Dictionary

ODS EXCLUDE Statement
Specifies output objects to exclude from ODS destinations.
Category: ODS: Output Control

Syntax

\[
\texttt{ODS <ODS-destination> EXCLUDE exclusion(s) | ALL | NONE;}
\]

Required Arguments

\textit{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” in \textit{SAS Output Delivery System: User’s Guide}.

Each \textit{exclusion} has the following form.

\textit{output-object} <\texttt{(PERSIST)}> 

\textit{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, the following is the full path of the output object:

  \texttt{Univariate.City_Pop_90.TestsForLocation}

- 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, suppose the full path is the following:

  \texttt{Univariate.City_Pop_90.TestsForLocation}

  Then the partial paths are as follows:

  \texttt{City_Pop_90.TestsForLocation}
  \texttt{TestsForLocation}

- a label that is enclosed in quotation marks.
For example:

"The UNIVARIATE Procedure" "Tests for Location"

- a label path. For example, the following is the label path for the output object:

"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, suppose the label path is the following:

"The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"

Then the partial label paths are as follows:

"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 “ODS TRACE Statement” on page 76.

(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.

Tips Using ODS EXCLUDE ALL is different from closing a destination. The destination remains open, but no output objects are sent to it.
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.

**Tips**

ODS EXCLUDE NONE has the same effect as ODS SELECT ALL.

To temporarily suspend a destination, use ODS SELECT NONE. Use ODS SELECT ALL when you want to resume sending output to the suspended destination.

**Optional Arguments**

**NOWARN**

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

**Requirement**

The NOWARN option must be enclosed in parentheses.

**Interaction**

The NOWARN option cannot be used with the ALL option or the NONE option.

**Example**

The ODS EXCLUDE statement in the following example specifies that no warning is created if the output object Summary is requested but not created.

```sas
ods exclude summary (nowarn);

proc contents data=sashelp.class;
run;
```

**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” in SAS Output Delivery System: User’s Guide.

**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” in SAS Output Delivery System: User’s Guide.

**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:

```sas
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 type 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 name 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>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

Details
Using 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.

**TIP** 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.

You can view the contents of the exclusion and selection lists by using the ODS SHOW statement. The current selection list 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” in SAS Output Delivery System: User’s Guide. For a list of ODS output destinations and explanations of each, see “Understanding ODS Destinations” in SAS Output Delivery System: User’s Guide.

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 written to 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 22, the “ODS EXCLUDE Statement” on page 14, and the “ODS TRACE Statement” on page 76.

**Example: Conditionally Excluding Output Objects and Sending Them to Different Output Destinations**

**Features:**
- ODS EXCLUDE statement:
  - Options: ODS-Destination, WHERE=
- ODS HTML statement options:
  - CONTENTS=
  - FRAME=
  - PAGE=
  - TEXT=
- ODS PDF statement options:
  - TEXT=
  - STARTPAGE=
- PROC UNIVARIATE

**Program**
```sas
options nodate;
data BPpressure;
  length PatientID $2;
  input PatientID $ Systolic Diastolic @@;
```

Program Description

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;

ods html text='Systolic Blood Pressure' file='Systolic-body.html'
   frame='Systolic-frame.htm'
   contents='Systolic-contents.htm'
   page='Systolic-page.htm';

ods pdf file='Diastolic.pdf' text='Diastolic Blood Pressure' startpage=no;

ods html exclude where=(_path_ ? "Diastolic") ;
ods pdf exclude where=(_path_ ? "Systolic") ;

proc univariate data=BPressure;
    var Systolic Diastolic;
run;
ods html close;
ods pdf close;

Create HTML output and add text.

ods html text='Systolic Blood Pressure' file='Systolic-body.html'
   frame='Systolic-frame.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 from the HTML destination output objects that have 'Diastolic' in the pathname. The second ODS EXCLUDE statement excludes from the PDF destination output objects that have 'Systolic' in the pathname.

    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. 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 cannot view the HTML file specified by the FRAME attribute until you close your SAS session.

    ods html close;

Close the PDF destination. This ODS PDF statement closes the PDF destination and all the files that are associated with it.

    ods pdf close;
Output

Output 3.1  Partial HTML Output with Systolic Output Objects

<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The Univariate Procedure</td>
</tr>
<tr>
<td>• Diastolic</td>
</tr>
<tr>
<td>• Moments</td>
</tr>
<tr>
<td>• Basic Measures of Location and Variability</td>
</tr>
<tr>
<td>• Tests For Location</td>
</tr>
<tr>
<td>• Quantiles</td>
</tr>
<tr>
<td>• Extreme Observations</td>
</tr>
</tbody>
</table>

Systolic Blood Pressure

<table>
<thead>
<tr>
<th>The SAS System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Moments</strong></td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>Sum Weights</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Sum Observations</td>
</tr>
<tr>
<td>Std Deviation</td>
</tr>
<tr>
<td>Variance</td>
</tr>
<tr>
<td>Skewness</td>
</tr>
<tr>
<td>Kurtosis</td>
</tr>
<tr>
<td>Uncorrected SS</td>
</tr>
<tr>
<td>Corrected SS</td>
</tr>
<tr>
<td>Coeff Variation</td>
</tr>
<tr>
<td>Std Error Mean</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Basic Statistical Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Location</strong></td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>Mode</td>
</tr>
<tr>
<td><strong>Variability</strong></td>
</tr>
<tr>
<td>Std Deviation</td>
</tr>
<tr>
<td>Variance</td>
</tr>
<tr>
<td>Range</td>
</tr>
<tr>
<td>Interquartile Range</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tests for Location: Mu0=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
</tr>
<tr>
<td>Student's t</td>
</tr>
</tbody>
</table>

Output 3.2  Partial PDF Output with Diastolic Output Objects

See Also

Statements

- "ODS SELECT Statement" on page 22
ODS SELECT Statement

Specifies output objects for ODS destinations.

Category: ODS: Output Control

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.

See: “ODS EXCLUDE Statement” on page 14

Example: “Example 7: Table Header and Footer Border Formatting” on page 500

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” in SAS Output Delivery System: User’s Guide. 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, the following is the full path of the output object:

  Univariate.City_Pop_90.TestsForLocation

- 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, suppose the full path is the following:

  Univariate.City_Pop_90.TestsForLocation

  Then the partial paths are as follows:

  City_Pop_90.TestsForLocation
TestsForLocation

- a label that is enclosed in quotation marks.
  For example:
  "The UNIVARIATE Procedure" "Tests For Location"

- a label path. For example, the label path for the output object is as follows:
  "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, suppose the label path is the following:
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"

  Then the partial label paths are as follows:
  "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  "ODS TRACE Statement" on page 76

(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.
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.

**Tips**

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.

To temporarily suspend a destination, use ODS SELECT NONE. Use ODS SELECT ALL when you want to resume sending output to the suspended destination.

**Optional Arguments**

**NOWARN**

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

**Requirement**
The NOWARN option must be enclosed in parentheses.

**Interaction**
The NOWARN option cannot be used with the ALL option or the NONE option.

**Example**
The ODS SELECT statement in the following example specifies that no warning is created if the output object Summary is requested but not created.

```sas
ods select summary (nowarn);
proc contents data=sashelp.class;
run;
```

**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.

**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” in SAS Output Delivery System: User's Guide.

**See**

**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:

```sas
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.

```sas
(subsetting-variable <comparison-operator where-expression-n> )
```
**Subsetting variables**

Subsetting variables are a special type 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 variables 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 name path of the output object.

**Operator**

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>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ¬= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**Details**

**Using 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.

**TIP** 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.

You can view the contents of the exclusion and selection lists by using the ODS SHOW statement. The current selection list 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" in SAS Output Delivery System: User’s Guide. For a list of ODS output destinations and explanations of each, see "Understanding ODS Destinations" in SAS Output Delivery System: User’s Guide.

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 written to 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 22, the "ODS EXCLUDE Statement" on page 14, and the "ODS TRACE Statement" on page 76.

**Examples:**

**Example 1: Using a Selection List with Multiple Procedure Steps**

*Features:*

ODS SELECT statement:
  - with label
  - with name
  - with and without PERSIST
  - ALL

ODS SHOW statement

ODS HTML statement options:
  - BODY=
  - CONTENTS=
  - FRAME=
  - PAGE=

PROC GLM
PROC PRINT
PROC PLOT
Details
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.

This example creates and prints data sets from the parameter estimates that PROC GLM generates. This procedure is part of SAS/STAT software.

Program
ods html body='odspersist-body.htm'
    frame='odspersist-frame.htm'
    contents='odspersist-contents.htm'
    page='odspersist-page.htm';
ods show;
ods select ParameterEstimates
    "Type III Model ANOVA";
ods show;
proc glm data=iron;
    model loss=fe;
    title 'Parameter Estimates and Type III Model ANOVA';
run;
ods show;
quit;
ods show;
proc glm data=iron;
    model loss=fe;
    title 'All Output Objects Selected';
run;
quit;
ods select OverallANOVA(persist) "Fit Statistics";
proc glm data=iron;
    model loss=fe;
    title 'OverallANOVA and Fitness Statistics';
run;
quit;
ods show;
proc glm data=iron;
    model loss=fe;
    title 'OverallANOVA';
    title2 'Part of the Selection List Persists';
run;
quit;
proc print data=iron;
    title 'The IRON Data Set';
run;
ods select all;
proc plot data=iron;
   plot fe*loss='*' / vpos=25;
   label fe='Iron Content'
       loss='Weight Loss';
   title 'Plot of Iron Versus Loss';
run;
quit;
ods html close;

Program Description

Create HTML output. The ODS HTML statement creates the body, contents, frame, and page files. 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 the “SAS Log” on page 31.

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 the “SAS Log” on page 31.

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 31. 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 the “SAS Log” on page 31.

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 the “SAS Log” on page 31.

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 31. 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. 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 31.

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 the “SAS Log” on page 31.

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 31.

```sas
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.

```sas
quit;
```

PROC PRINT does not produce any output that is named OverallANOVA. Therefore, no PROC PRINT output is sent to the ODS destinations.

```sas
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.

```sas
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).

```sas
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.

```sas
quit;
```

Close the HTML destination. This ODS HTML statement closes the HTML destination and all the files that are associated with it.

```sas
ods html close;
```
SAS Log

Output 3.3 The ODS SHOW Statement Writes the Current Selection List to the SAS Log.

Example 1: Using a Selection List with Multiple Procedure Steps

```
0  ods html body='odspersist-body.htm' frame='odspersist-frame.htm' content='odspersist-contents.htm' page='odspersist-page.htm';
1  ods show;
2  ods select ParameterEstimates Type Iii Model ANOVA;
3  proc glm data=iron;
4  model loss=f0;
5  title 'Parameter Estimates and Type Iii Model ANOVA';
6  run;
7  ods show;
8  ods select All;
9  proc glm data=iron;
10  model loss=f0;
11  title 'All Output Objects Selected';
12  run;
13  ods select OverallANOVA(persist) "Fit Statistics";
14  proc glm data=iron;
15  model loss=f0;
16  title 'OverallANOVA and Fitness Statistics';
17  run;
18  ods show;
19  proc glm data=iron;
20  model loss=f0;
21  title 'Part of the Selection List Persists';
22  run;
23  quit;
```

HTML Output

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

Features:

- ODS SELECT statement option:
  
  WHERE=

- ODS TRACE statement options:
  
  LABEL
  EXCLUDED

- ODS HTML statement

PROC UNIVARIATE

Program

data BPressure;
  length PatientID $2;
  input PatientID $ Systolic Diastolic @@;
datalines;
  CK 120 50  SS 96  60  FR 100 70
Program Description

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;

    title 'Systolic and Diastolic Blood Pressure';
ods trace on / label excluded;
ods select where=( _path_ ? "Diastolic" and _name_='Moments');
proc univariate data=BPressure;
    var Systolic Diastolic;
    run;

Add a title.

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 about 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 pathname.

ods select where=( _path_ ? "Diastolic" and _name_='Moments');

Create the output objects and send the selected output objects to the open destination. 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 destination.

proc univariate data=BPressure;
    var Systolic Diastolic;

run;

SAS Log: Trace Record

Output 3.5  SAS Log Including Trace Record

data BPDetails;
length PatientID $2;
input PatientID Systolic Diastolic @@;
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.BPRESSURE has 22 observations and 3 variables.
NOTE: DATA statement used (Total process time):  
   real time 0.23 seconds  
   cpu time 0.07 seconds

run;

title 'Systolic and Diastolic Blood Pressure'; ods trace on / label excluded;
ods select where(_path_ != "Systolic" and _name_="Moments");
proc univariate;
   dateBPDetails;
NOTE: writing HTML Body file: ssahtml.htm
   var Systolic Diastolic;
run;

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'
See Also

Statements
- “ODS EXCLUDE Statement” on page 14
- “ODS TRACE Statement” on page 76
Other SAS Statements That Are Used with ODS Procedures

Dictionary

ODS GRAPHICS Statement

Enables or disables ODS Graphics processing and sets graphics environment options. This statement affects ODS template-based (ODS Graphics) graphics only. The ODS GRAPHICS statement does not affect device-based graphics (SAS/GRAPH).

Valid in: Anywhere
Category: ODS: Output Control
Default: ON. Beginning in SAS 9.4, ODS Graphics is enabled by default on all platforms except z/OS. When running SAS in batch mode, the default is 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.
See: For information about common tasks for managing ODS Graphics output, see SAS Graph Template Language: User’s Guide.

Syntax

ODS GRAPHICS <OFF | ON> </options> ;
Summary of Optional Arguments

**ANTIALIAS | NOANTIALIAS | ANTIALIAS= ON | OFF**
specifies whether anti-aliasing is applied to the rendering of the line and markers in any graph.

**ANTIALIASMAX= n**
specifies the maximum number of graphics elements before anti-aliasing is disabled.

**ATTRPRIORITY=COLOR | NONE**
specifies a priority for cycling of the group attributes.

**BORDER | NOBORDER | BORDER=ON | OFF**
specifies whether to draw a border around each graph.

**BYLINE=NOBYLINE | TITLE | FOOTNOTE**
specifies how the BY line is displayed in graphs.

**DATASKINMAX= n**
specifies the maximum number of graphical elements allowed per plot when a data skin is applied.

**DISCRETEMAX= n**
specifies the maximum number of discrete values to be shown in any graph.

**DRILLTARGET="_blank" | "_self" | "_parent" | "_top" | "frame-name"**
specifies the window that displays the drill-down output.

**GROUPMAX= n**
specifies the maximum number of group values to be shown in any graph.

**HEIGHT=dimension**
specifies the height of a graph.

**IMAGEMAP | NOIMAGEMAP | IMAGEMAP=ON | OFF**
specifies whether data tips are generated.

**IMAGENAME="filename"**
specifies the base image filename.

**LABELMAX= n**
specifies the maximum number of labeled areas before data label collision avoidance is disabled.

**LABELPLACEMENT= GREEDY | SA**
specifies the label-placement algorithm to use for positioning labels in the graphs.

**LEGENDAREAMAX= n**
specifies an integer that is interpreted as the maximum percentage of the overall graphics area that a legend can occupy.

**LINEPATTERNOBSMAX= n**
specifies the maximum the number of vertices for a patterned line.

**LOESSOBSMAX= n**
specifies an upper limit for the number of observations that can be used with a loess plot.

**NBINSMAX= n**
specifies the maximum number of bins that are processed for histograms.

**NXYBINSMAX= n**
specifies the maximum number of bins that are processed for heat maps.

**OBSMAX= n**
specifies the maximum number of observations that are processed.

**OUTPUTFMT= file–type | STATIC**
specifies the output format used to generate image or vector graphic files.

**PANELCELLMAX=n**

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

**PUSH**

**POP**

pushes and pops ODS GRAPHICS settings in a stack.

**RESET | RESET= option**

Reset one or more ODS GRAPHICS options to its default.

**SCALE | NOSCALE | SCALE=ON | OFF**

specifies whether the content of a graph is scaled proportionally.

**SCALEMARKERS | NOSCALEMARKERS | SCALEMARKERS=ON | OFF**

specifies whether the plot markers are to be scaled with the graph size.

**SHOW**

writes the current ODS Graphics settings to the SAS log.

**STACKDEPTHMAX=n**

specifies the maximum stack depth for PUSH and POP requests.

**SUBPIXEL | NOSUBPIXEL | SUBPIXEL=ON | OFF**

specifies whether subpixel rendering should be used for rendering ODS Graphics.

**TIPMAX=n**

specifies the maximum number of distinct data tip areas allowed before data tips are disabled.

**TOTALCELLMAX=n**

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

**WIDTH=dimension**

specifies the width of any graph.

### Without Arguments

If ODS Graphics is currently disabled, then specifying the ODS GRAPHICS statement without options enables it. If ODS Graphics is currently enabled, then specifying the ODS GRAPHICS statement leaves it enabled.

### Required Arguments

**ON**

enables ODS Graphics processing. This is the default if no argument is used.

---

**Note:** Beginning in SAS 9.4, ODS Graphics is enabled by default on all platforms except z/OS.

---

**OFF**

disables ODS Graphics processing.

---

**Alias** YES

**Alias** NO
Optional Arguments

**ANTIALIAS | NOANTIALIAS | ANTIALIAS= ON | OFF**
specifies whether anti-aliasing is applied to the rendering of the line and markers in any graph. Anti-aliasing smooths the appearance of 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**
smooths jagged edges of all components in the graph.

**NOANTIALIAS**
does not smooth jagged edges of components other than text in the graph.

**ANTIALIAS=ON | OFF**
specifies whether anti-aliasing is applied to the rendering of the line and markers in the graph.

**ON**
smooths jagged edges of all components in the graph.

Alias YES

**OFF**
does not smooth jagged edges of components other than text in the graph.

Alias NO

Default ANTIALIAS or ANTIALIAS=ON | YES

Restriction If the number of markers or lines in the plot exceeds the number specified by the ANTIALIASMAX= option, then the ANTIALIAS option is disabled. This is true even if you specify the option ANTIALIAS=ON or ANTIALIAS.

**ANTIALIASMAX= n**
specifies the maximum number of graphics elements before anti-aliasing is disabled. For example, if there are more than 400 scatter point markers to be anti-aliased and ANTIALIASMAX=400, then no markers are anti-aliased. The default value is 4000.

Note: Prior to SAS 9.4M3, the ANTIALIASMAX= option specifies the maximum number of observations in the graph data to be anti-aliased before anti-aliasing is disabled. The default is 4000. When the graph data contains more than 4000 observations, anti-aliasing is disabled for the entire graph. Starting with SAS 9.4M3, the ANTIALIASMAX= option specifies the maximum number of graphics elements to be anti-aliased in each plot on a per-plot basis. The default remains at 4000. If any plot in a graph contains more than 4000 elements, anti-aliasing is disabled for that plot. Anti-aliasing is enabled for the rest of the graph in that case.

**n** specifies a positive integer.

Default 4000
ATTRPRIORITY=COLOR | NONE
specifies a priority for cycling of the group attributes.

COLOR
assigns priority to the color attribute rotation by cycling through the list of colors while holding the marker symbol and line pattern constant. When all of the colors are exhausted, the marker symbol and line style attributes increment to the next element, and then the colors in the list are repeated. This pattern repeats as needed.

NONE
does not use an attribute priority in the rotation pattern, even if one is set in the active style’s AttrPriority attribute. The rotation pattern cycles progressively through the attribute lists.

Default
The AttrPriority attribute of the graph style element, or NONE if the current style does not define the AttrPriority style attribute.

Interaction
The default lists of data colors, contrast colors, marker symbols, and line patterns are set in the active style’s GraphData1–GraphDataN elements.

Tip
Use the ATTRPRIORITY=NONE option if you want groups to be distinguished by color, marker, and line changes for all styles that use color.

BORDER | NOBORDER | BORDER=ON | OFF
specifies whether to draw a border around each graph.

BORDER
draws a border around the graph.

NOBORDER
does not draw a border around the graph.

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

ON
draws a border around the graph.

Alias YES

OFF
does not draw a border around the graph.

Alias NO

Default BORDER or BORDER=ON | YES

BYLINE=NOBYLINE | TITLE | FOOTNOTE
specifies how the BY line is displayed in graphs. The option specifies how the BY line is displayed when an analysis is run with a BY statement. By default, no BY line is displayed.

The following code is an example of how the placement of the BY line is controlled in most graph templates:

if (_BYTITLE_)
   entrytitle _BYLINE_ / textattrs=GraphValueText;
else
You can modify the graph template if you want to change how the BY line is displayed. Because most graphs have titles and few graphs have footnotes, the BY line looks better when it is displayed as a footnote. For complete documentation about the Graph Template Language, see SAS Graph Template Language: User’s Guide.

When the BYLINE= option is specified, and there are BY groups, ODS creates a BY line and sets the appropriate special dynamic variables. The following table lists the special dynamic variables for BY lines. For complete documentation about special dynamic variables, see “Special Dynamic Variables” in SAS Graph Template Language: User’s Guide.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>BYFOOTNOTE</em></td>
<td>This variable is set to 1 when you specify a BY statement and the ODS GRAPHICS BYLINE= option is set to FOOTNOTE. Otherwise, the variable is set to 0 or is NULL.</td>
</tr>
<tr>
<td><em>BYTITLE</em></td>
<td>This variable is set to 1 when you specify a BY statement and the ODS GRAPHICS BYLINE= option is set to TITLE. Otherwise, the variable is set to 0 or is NULL.</td>
</tr>
<tr>
<td><em>BYLINE</em></td>
<td>This variable contains the text string that can be displayed as a title or footnote.</td>
</tr>
</tbody>
</table>

The variables in the table are set automatically only for analytical procedures that support ODS Graphics. For all other procedures, the variables are not set automatically (NULL). To determine whether the procedure you are using supports ODS Graphics, refer to the procedure documentation.

- **NOBYLINE** specifies that no BY line is displayed. NOBYLINE is the default.
- **FOOTNOTE** specifies that the BY line is displayed as a left-justified graph footnote. This is the recommended setting.
- **TITLE** specifies that the BY line is displayed as a centered graph title. Specifying TITLE is not recommended because graphs are not designed to have additional title lines.

**Default** NOBYLINE

**Restriction** This option does not work with the ODS Graphics procedures such as SGPLOT and SGPANEL. To remove BY lines in those procedures, use the NOBYLINE SAS system option. Example: options nobyline;

**DATASKINMAX=n** specifies the maximum number of graphical elements allowed per plot when a data skin is applied.
Note: This feature applies to SAS 9.4M1 and later releases.

\[ n \]

specifies a positive integer.

Default 200

**DISCRETEMAX=\( n \)**

specifies the maximum number of discrete values to be shown in any graph. Bar charts 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

**Tips**

Some plot layers might be unaffected by the DISCRETEMAX= option, and those layers are rendered. If all layers are affected, a blank graph is rendered.

If the value specified by the DISCRETEMAX= option is exceeded by any plot layer in the graph, that layer is not drawn and a warning message is issued. In that case, use the DISCRETEMAX= option to increase the maximum number of discrete values that are allowed. Starting with SAS 9.4M5, the log message includes a suggested value for DISCRETEMAX=.

**DRILLTARGET=’_blank’ | ’_self’ | ’_parent’ | ’_top’ | ’frame-name’**

specifies the window that displays the drill-down output.

Note: This option is supported only for HTML.

’_blank’

opens a new browser window to display the drilldown output.

Default _blank is the default.

Requirements You must enclose _blank in quotation marks.

You must specify _blank in lowercase.

’_self’

opens the drill-down output in the same window.

Requirements You must enclose _self in quotation marks.

You must specify _self in lowercase.

’_parent’

opens the drill-down output in the parent frame.

Requirements You must enclose _parent in quotation marks.

You must specify _parent in lowercase.
"_top"
opens the drill-down output in the full body of the window.

Requirements
You must enclose _top in quotation marks.

You must specify _top in lowercase.

"frame-name"
opens the drill down output in the named frame in the current window. If the name does not exist, the output is opened in a new window.

Requirement
You must enclose frame–name in quotation marks.

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 is rendered. The system ignores the GROUP= option and issues a warning message. In that case, use the GROUPMAX= option to increase the maximum number of group values that are allowed. Starting with SAS 9.4M5, the log message includes a suggested value for GROUPMAX=.

HEIGHT=dimension
specifies the height of a graph.

dimension
is a nonnegative number followed by one of these units of measure:

Table 4.2 Units of Measure for Dimension

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pct or %</td>
<td>Percentage</td>
</tr>
<tr>
<td>pt</td>
<td>Point size (72 points = 1 inch)</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

Defaults
The value of the SAS registry entry "ODS > ODS GRAPHICS > Design Height" or the value of the DesignHeight= option in a STATGRAPH template. Typically, the value is 480px.

For the PRINTER destination, units of 1/150 of an inch are used by default.
Tips
If only the HEIGHT= option is specified, then the default aspect ratio of the
graph is maintained.

When you change the size of a graph, by default graph elements such as
markers, lines, text, and titles are scaled proportionally. This scaling takes
effect even when you specify dimensions for the graph elements. To
disable the scaling, specify the NOSCALE option.

IMAGEMAP | NOIMAGEMAP | IMAGEMAP=ON | OFF
controls data tips and drill down generation. Data tips are pieces of explanatory
text that appear when you hold the mouse pointer over the data portions of a
graph contained in an HTML page.

IMAGEMAP
specifies to generate data tips.

NOIMAGEMAP
specifies not to generate data tips.

IMAGEMAP= ON | OFF
controls data tips generation.

ON
specifies to generate data tips.

Alias YES

OFF
specifies not to generate data tips.

Alias NO

Default NOIMAGEMAP or IMAGEMAP=OFF | NO

Restrictions This option applies only when one of the ODS HTML* destinations
is used.

Prior to SAS 9.4M5, an image map is not generated using SVG with
ODS Graphics. The image map data that is used to produce tooltips
and links is written directly in the SVG and is not part of the HTML.
Using HTML5 with the inline SVG mode (the default value), the
tooltips and links are written in the SVG portion of the document. (If
you are using SAS 9.4M5, see the following Note.)

Interaction When IMAGEMAP | IMAGEMAP=ON is specified and the ODS
HTML destination is used, the IMAGE_DPI option in the ODS
HTML destination is ignored, if specified, and the default image
resolution of 96 DPI is used.

Note Starting with SAS 9.4M5, image maps are supported with SVG
output using HTML5. However, image maps are supported only
when the HTML5 SVG mode is INLINE (the default value).

IMAGENAME="filename"
specifies the base image filename. If more than one image is generated, each is
assigned a filename that consists of 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 OUTPUTFMT= option.

**Note:** Starting with SAS 9.4M5, if a BY statement is in effect, you can uniquely name image files based on BY groups. To do this, insert #BY text into the image name. For more information, see “Substituting BY Line Values in the Image Name” on page 59.

**Default**

The name of the output object.

**Restriction**

filename must be a single name. It must not include any path specification or image-format name extension.

**Requirement**

You must enclose filename in quotation marks.

**See**

“Specifying and Resetting the Image Name” on page 59

**LABELMAX=n**

specifies the maximum number of labeled areas before data label collision avoidance is disabled. For example, if there are more than 50 points to be labeled and LABELMAX=50, then collision avoidance is turned off and the labels are all displayed at the top right of the data points.

\[ n \]

specifies a positive integer.

**Default**

200

**Restriction**

Data label collision avoidance is turned off under the following conditions:

- The number of observations with nonmissing labels exceeds the value specified by LABELMAX=.
- The number of observations exceeds five times the value specified by LABELMAX=.

A message is then sent to the SAS log.

**Tip**

To turn off collision avoidance, specify LABELMAX=0.

**LABELPLACEMENT= GREEDY | SA**

specifies the label-placement algorithm to use for positioning labels in the graphs. The following labels are affected:

- data labels for needle plots, scatter plots, series plots, step plots, and vector plots
- vertex labels for line charts
- curve labels when the curve label is positioned at the start or end of the curve

**GREEDY**

specifies the Greedy method for managing label collision. The Greedy method tries different placement combinations in order to find an optimal approximation that avoids collisions. Label placement using this method is often less optimal than label placement using the Simulated Annealing (SA) method. However, depending on the number of data points and the potential for label collisions, the Greedy process can be significantly faster.
SA specifies the Simulated Annealing method for managing label collision. The SA method attempts to determine the global minimization-of-cost function, which is based on a simulated annealing algorithm. The resulting label placement is usually better than placement using the Greedy method. However, depending on the number of data points and the potential for label collisions, the SA method can be significantly slower.

Restriction For BANDPLOT and LINECHART, the SA method has no effect on the curve labels when the CURVELABELPOSITION= option specifies START or END.

LEGENDAREAMAX=\(n\)
specifies an integer that is interpreted as the maximum percentage of the overall graphics area that a legend can occupy.

Note: Starting with SAS 9.4M3, LEGENDAREAMAX= replaces MAXLEGENDAREA=. However, MAXLEGENDAREA= is supported as an alias. It is recommended that you use LEGENDAREAMAX=.

\(n\)
specifies a positive integer.

Alias MAXLEGENDAREA=

Default 20

Range 0–100

Tip To turn off the legend, specify LEGENDAREAMAX=0. No warning is issued when the legend is turned off in this way.

LINEPATTERNOBSMAX=\(n\)
specifies the maximum the number of vertices for a patterned line. If the number of vertices exceeds the specified limit, the plot is not drawn.

\(n\)
specifies a positive integer.

Note: This feature applies to SAS 9.4M5 and later releases.

Default 10000

Notes If the number of vertices exceeds the specified limit, the plot is not drawn and a note is written to the SAS log. In that case, increase LINEPATTERNOBSMAX= to the value suggested in the log message, or change the line pattern to SOLID.

This option applies only to graphs created with the Graph Template Language and with the ODS Graphics procedures.

LOESSOBSMAX=\(n\)
specifies an upper limit for the number of observations that can be used with a loess plot.
Note: Starting with SAS 9.4M3, LOESSOBSMAX= replaces LOESSMAXOBS=. However, LOESSMAXOBS= is supported as an alias. It is recommended that you use LOESSOBSMAX=.

If the number of observations of the loess plot exceeds the specified limit, the loess plot is not drawn.

For example, the following specifies that the most observations a loess plot can have is 1000.

**LOESSOBSMAX=1000**

Alias LOESSMAXOBS=

Default 5000

**NBINSMAX=**

specifies the maximum number of bins that are processed for histograms. This option affects both computed and parameterized histograms, when available.

**Note:** This feature applies to SAS 9.4M4 and later releases.

\[ n \]

specifies a positive integer.

Default 10000

Interaction If you specify the number of bins in the histogram plot statement, that option is honored regardless of the NBINSMAX= option.

**NXYBINSMAX=**

specifies the maximum number of bins that are processed for heat maps. This option affects both computed and parameterized heat maps. The option applies to the product of the X and Y bins.

**Note:** This feature applies to SAS 9.4M4 and later releases.

\[ n \]

specifies a positive integer.

Default 100000

Interaction If you specify the number of bins in the heat map statement, that option is honored regardless of the NXYBINSMAX= option.

**OBSMAX=**

specifies the maximum number of observations that are processed. If the number of observations in the data set exceeds the value specified for OBSMAX=, the procedure step terminates with a log message.

\[ n \]

specifies a positive integer.

Alias MAXOBS=

Default 2 million observations
There are other ways to control the number of observations: `CASDATALIMIT=` system option, `DATALIMIT=` option in the CAS LIBNAME statement, and `DATALIMIT=` data set option. If the CAS data transfer limit is set lower than `OBSMAX=`, then `OBSMAX=` has no effect.

**OUTPUTFMT=** *file-type* | **STATIC**

specifies the output format used to generate image or vector graphic files. If the image or vector graphic format is not valid for the active output destination, the format is automatically changed to the default format for that destination.

*file-type*

is the image or vector graphic format to be generated. See “Supported File Types for Output Destinations” on page 64.

**STATIC**

uses the best quality static image format for the active output destination. This is the default output format.

TIP The STATIC keyword can be used to reset the output format to its default state.

Default **STATIC**

See “Specifying the Image Format” on page 61

**PANELCELLMAX=** *n*

specifies the maximum number of cells in a graph panel where the number of cells is determined dynamically by classification variables. If the number of cells in the panel exceeds the specified limit, the panel is not drawn.

*n*

specifies a positive integer.

Default **1000**

Note Graphs with DataPanel or DataLattice templates layouts are affected. In the ODS Graphics Procedures, this option affects graphs that are created with the SGPANEL procedure. If the value specified by the `PANELCELLMAX=` option is exceeded by any of these layouts, an empty graph is rendered and a warning message is issued.

Tip You can use the `TOTALCELLMAX=` option to control the total maximum number of cells in the graph.

**PUSH | POP**

pushes and pops ODS GRAPHICS settings in a stack. This feature enables you to temporarily save your custom settings in a stack and later restore those settings.

Note: This feature applies to SAS 9.4M3 and later releases.

**PUSH**

pushes the current ODS GRAPHICS settings to a stack.
POP
restores the most recently pushed settings from the stack. For each PUSH action, you can specify a POP request. ODS issues a warning if you specify POP without a corresponding PUSH. In that case, nothing is popped because nothing has been pushed.

The pushed settings remain in the stack in the current SAS session until they are popped or the stack is emptied.

Interaction You can specify PUSH as many times as you like up to the limit that is defined by the STACKDEPTHMAX= option. You can also use STACKDEPTHMAX= to empty the stack. For more information, see “Managing the Stack Depth” on page 68.

Note Order of specification is important when using the PUSH and POP options. For more information, see “About PUSH and POP” on page 66.

Tip Use the SHOW option to show the current ODS GRAPHICS settings.

See “Temporarily Saving and Restoring ODS GRAPHICS Settings” on page 66

---

**RESET | RESET= option**
Reset one or more ODS GRAPHICS options to its default.

**RESET**
resets all options to their defaults.

**RESET=**
resets one of the following to its default:

- **ALL**
  resets all reset-options to their defaults.

- **ANTIALIAS**
  resets the ANTIALIAS= option to its default.

  See **ANTIALIAS= on page 40**

- **ANTIALIASMAX**
  resets the ANTIALIASMAX= option to its default.

  See **ANTIALIASMAX= on page 40**

- **ATTRPRIORITY**
  resets the ATTRPRIORITY= option to its default.

  See **ATTRPRIORITY= on page 41**

- **BORDER**
  resets the BORDER= option to its default.

  See **BORDER= on page 41**

- **BYLINE**
  resets the BYLINE= option to its default.

  See **BYLINE= on page 41**
DATASKINMAX
resets the DATASKINMAX= option to its default.

See \texttt{DATASKINMAX=} on page 42

DISCRETEMAX
resets the DISCRETEMAX= option to its default.

See \texttt{DISCRETEMAX=} on page 43

DRILLTARGET
resets the DRILLTARGET= option to its default.

See \texttt{DRILLTARGET=} on page 43

GROUPMAX
resets the GROUPMAX= option to its default.

See \texttt{GROUPMAX=} on page 44

HEIGHT
resets the HEIGHT= option to its default.

See \texttt{HEIGHT=} on page 44

IMAGEMAP
resets the IMAGEMAP= option to its default.

Note: Not all output destinations support this feature.

See \texttt{IMAGEMAP=} on page 45

IMAGENAME
resets the IMAGENAME= option to its default.

Note: This feature applies to \texttt{SAS 9.4M3} and later releases.

See \texttt{IMAGENAME=} on page 45

INDEX <(positive-integer)>  
resets the index counter that is appended to static image files.

When specifying this option, you can also specify the value for the index counter. The number that you specify must be enclosed in parentheses. \texttt{positive-integer} determines the suffix for the next subsequent image, and increments with each new image. This feature applies to \texttt{SAS 9.4M3} and later releases.

See \texttt{"Resetting the Image Name"} on page 59

LABELMAX
resets the LABELMAX= option to its default.

See \texttt{LABELMAX=} on page 46

LABELPLACEMENT
specifies the label-placement algorithm to use for positioning labels in the graphs.
LEGENDAREAMAX
resets the LEGENDAREAMAX= option to its default.
See LEGENDAREAMAX= on page 47

LOESSOBSMAX
resets the LOESSOBSMAX= option to its default.
See LOESSOBSMAX= on page 47

NBINSMAX
resets the NBINSMAX= option to its default.
See NBINSMAX= on page 48

NXYBINSMAX
resets the NXYBINSMAX= option to its default.
See NXYBINSMAX= on page 48

OUTPUTFMT
resets the OUTPUTFMT= option to its default.

Note: This feature applies to SAS 9.4M3 and later releases.
See OUTPUTFMT= on page 49

PANELCELLMAX
resets the PANELCELLMAX= option to its default.
See PANELCELLMAX= on page 49

SCALE
resets the SCALE= option to its default.
See SCALE= on page 53

SCALEMARKERS
resets the SCALEMARKERS= option to its default.
See SCALEMARKERS= on page 53

STACKDEPTHMAX
resets the STACKDEPTHMAX= option to its default.

Note: This feature applies to SAS 9.4M3 and later releases.
See STACKDEPTHMAX= on page 55

SUBPIXEL
resets the SUBPIXEL option to its default.
Note: This feature applies to SAS 9.4M3 and later releases.

See SUBPIXEL on page 56

TIPMAX
reset the TIPMAX= option to its default.

See TIPMAX = on page 56

WIDTH
reset the WIDTH= option to its default.

See WIDTH= on page 57

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

SCALE
scales the components of a graph proportionally.

NOSCALE
does not scale the components of a graph proportionally.

SCALE=ON | OFF
specifies whether the content of a graph is scaled proportionally.

ON
scales the components of a graph proportionally.

Alias YES

OFF
does not scale the components of a graph proportionally.

Aliases NOSCALE

NO

Default SCALE or SCALE=ON | YES

SCALEMARKERS | NOSCALEMARKERS | SCALEMARKERS=ON | OFF
specifies whether the plot markers are to be scaled with the graph size. The scaling factor is based on the height of the graph cells and the height of the graph.

SCALEMARKERS
scales the markers with the graph size.

NOSCALE
does not scale the markers with the graph size.

SCALEMARKERS=ON | OFF
specifies whether the plot markers are to be scaled with the graph size.

ON
scales the markers with the graph size.

Alias YES
OFF

does not scale the markers with the graph size.

Aliases NOSCALE

NO

Default SCALEMARKERS or SCALEMARKERS=ON | YES

Restriction Scaling is done only if the graph contains multiple cells or single nested cells.

SHOW

writes the current ODS Graphics settings to the SAS log. This option enables you to verify which settings are in effect. The option is especially useful when you use the PUSH and POP options to restore settings. For more information, see “Temporarily Saving and Restoring ODS GRAPHICS Settings” on page 66.

Note: This feature applies to SAS 9.4M3 and later releases.

If no options have been specified, then SHOW lists those options for which ODS currently knows the default values.

The following statement resets all settings and shows the default values.

ods graphics / reset=all show;

Here are the default values displayed in the SAS log:

ODS Graphics Settings
----------------------
Output format: STATIC
By line: NOBYLINE
Antialias: ON
Maximum Loess observations: 5000
Maximum stack depth: 1024
Stack depth: 0
MaxObs: 2000000
Maximum Histogram Bins: 10000
Maximum Heatmap Bins: 100000
Maximum Obs for Patterned Lines: 10000
Maximum Total Cells per BY-group: 2000

If you have specified the settings for one or more options, then SHOW includes those settings along with the defaults.

Order of specification is important when using the SHOW option. For example, the following statement shows the current settings and then sets the NOBORDER option.

ods graphics / show noborder;

However, the following statement sets the NOBORDER option and then shows the settings. The NOBORDER setting is shown in the log along with the other settings that are in effect.

ods graphics / noborder show;

The following statement resets all settings. It then sets the image width and shows the default settings along with the specified width.
Here are the default values plus the image width, as displayed in the SAS log:

```
ODS Graphics Settings
----------------------
Output format: STATIC
By line: NOBYLINE
Antialias: ON
Maximum Loess observations: 5000
Image width: 5in
Maximum stack depth: 1024
Stack depth: 0
MaxObs: 2000000
Maximum Histogram Bins: 10000
Maximum Heatmap Bins: 100000
Maximum Total Cells per BY-group: 2000
```

Tip If you have specified the settings for some options but want to see the default values without losing your specified settings, issue the following two statements. The first statement pushes your specified settings, resets all settings, and then lists options for which ODS currently knows the default values. The second statement restores your previous settings.

```
ods graphics / push reset=all show;
```

```
ods graphics / pop;
```

**STACKDEPTHMAX=n**

specifies the maximum stack depth for PUSH and POP requests. The stack is used to temporarily store ODS GRAPHICS settings when you issue PUSH requests. PUSH saves the current settings to the stack and increments the stack depth. POP restores the most recently saved settings from the stack and decrements the stack depth.

**Note:** This feature applies to **SAS 9.4M3** and later releases.

**n**

specifies a positive integer.

If n is less than the current stack depth, then the stack is popped until its depth equals n. Popping the stack does not affect other option settings.

**Defaults**

1024 is the default maximum depth

0 is the default depth

**Tips**

To empty the stack and then reset it to the default maximum depth, issue the following statement:

```
ods graphics / stackdepthmax=0 reset=stackdepthmax;
```

You can use any of the following commands to reset the stack to its default maximum depth:

```
reset=stackdepthmax
reset=all
reset
stackdepthmax=1024
```
SUBPIXEL | NOSUBPIXEL | SUBPIXEL=ON | OFF
specifies whether subpixel rendering should be used for rendering ODS Graphics. Subpixel rendering produces smoother curves and more precise bar spacing.

SUBPIXEL
always uses subpixel rendering, when applicable, for rendering lines and bars.

NOSUBPIXEL
never uses subpixel rendering.

SUBPIXEL=ON | OFF
specifies whether subpixel rendering should be used.

ON
always uses subpixel rendering, when applicable, for rendering lines and bars.

Alias YES

OFF
never uses subpixel rendering.

Alias NO

Default Subpixel rendering is always enabled for vector-graphics output. It is enabled by default for image output, unless the graph contains a scatter plot or a scatter-plot matrix. In those cases, subpixel rendering is disabled by default.

Requirement Antialiasing must be enabled for this option to have any effect. Antialiasing is enabled by default. To re-enable antialiasing, use the ANTIALIAS=ON option in the ODS GRAPHICS statement.

Tip For a large amount of data, antialiasing is disabled when the number of observations exceeds the default maximum of 4000 observations. In that case, subpixel rendering is also disabled. To increase the maximum, use the ANTIALIASMAX= option in the ODS GRAPHICS statement.

Note: Antialiasing must be enabled for this option to have any effect. Antialiasing is enabled by default. To re-enable antialiasing, use the ANTIALIAS=ON option in the ODS GRAPHICS statement.


TIPMAX=n
specifies the maximum number of distinct data tip 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 appear. The default maximum value is 500.

Note: Prior to SAS 9.4M3, the TIPMAX= option specifies the maximum number of observations in the graph data to be allowed before data tips are disabled. The default is 500. When the graph data contains more than 500 observations, data tips are disabled for the entire graph. Starting with SAS 9.4M3, the TIPMAX= option specifies the maximum number of data-tip areas allowed before
data tips are disabled. This threshold is applied separately for each plot. The default remains at 500. If any plot in a graph contains more than 500 data-tip areas, data tips are disabled for that plot. Data tips are enabled for the remaining plots in the graph.

\[ n \]

specifies a positive integer.

Default 500

**TOTALCELLMAX=** \( n \)

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

Note: This feature applies to SAS 9.4M5 and later releases.

If the number of cells exceeds the specified limit, the graph is not drawn. This option is useful when multiple classification variables or large amounts of data result in a large number of cells.

\[ n \]

specifies a positive integer.

Default 2000

Notes

If the number of cells exceeds the limit specified for TOTALCELLMAX=, the graph is not drawn and an error message is written to the SAS log. The error message includes a suggested value for TOTALCELLMAX=.

You can use the TOTALCELLMAX= option to increase the maximum number of cells. Note, however, that the processing time to render the graph increases with an increase in the maximum number of cells.

Graphs with DataPanel or DataLattice templates layouts are affected. In the ODS Graphics Procedures, this option affects graphs that are created with the SGPANEL procedure.

Tip

You can use the PANELCELLMAX= option to control the maximum number of cells in a graph panel.

**WIDTH=** \( \text{dimension} \)

specifies the width of any graph.

\( \text{dimension} \)

is a nonnegative number followed by one of these units of measure:

<table>
<thead>
<tr>
<th>Table 4.3 Units of Measure for Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
</tr>
<tr>
<td>in</td>
</tr>
<tr>
<td>mm</td>
</tr>
<tr>
<td>pct or %</td>
</tr>
<tr>
<td>pt</td>
</tr>
</tbody>
</table>
Defaults  The value of the SAS registry entry "ODS > ODS GRAPHICS > Design Width" or the value of the DesignWidth= option in a STATGRAPH template. Typically, this value is 640px.

For the PRINTER destination, units of 1/150 of an inch

Tips  If only the WIDTH= option is specified, then the default aspect of the graph is maintained.

When you change the size of a graph, by default graph elements such as markers, lines, text, and titles are scaled proportionally. This scaling takes effect even when you specify dimensions for the graph elements. To disable the scaling, specify the NOSCALE option.

Details
Using the ODS GRAPHICS Statement
You can enable ODS Graphics by using one 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: ODS Graphics is ON by default for these procedures: SGDESIGN, SGMAP, SGPANEL, SGPIE, SGPLOT, SGRENDER, and SGSCATTER. For other products, the initial state of 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.
Specifying and Resetting the Image Name

Specifying the Image Name

For ODS Graphics output, by default, the ODS object name is used as the "root" name for the image output file. The following example creates a GIF image named REGPLOT:

    ods graphics / imagename="regplot" outputfmt=gif;

The assigned name REGPLOT is treated as a "root" name and the first output created is named REGPLOT. Subsequent graphs are named REGPLOT1, REGPLOT2, and so on, with an increasing index counter. This numbering can be reset with the RESET=INDEX option.

Resetting the Image Name

The RESET=INDEX option enables you to reset the filename numbering sequence. This feature applies to SAS 9.4M3 and later releases.

For a usage example, suppose that you are developing a paper or a presentation and it takes several submissions to get the desired output. You can use the RESET or RESET=INDEX option to force each output to replace itself:

    ods graphics / reset=index ... ;

This specification causes all subsequent images to be created with the default or current image name.

When specifying this option, you can also specify the value for the index counter. The value that you specify determines the suffix for the next subsequent image. For example:

    ods graphics / reset=index(100) imagename="MyName";

The next graph that you produce is named MYNAME100.

This feature is useful for creating animated graphics. For example, for a sequence of 100 images, you might begin with the following statement:

    ods graphics / reset=index(1) imagename="MyName";

In the example, your program produces 100 images named MYNAME1, MYNAME2, ..., MYNAME100. If you later add more images to the animation, you might submit the following:

    ods graphics / reset=index(101) imagename="MyName";

The next generated image is named MYNAME101.

Substituting BY Line Values in the Image Name

Note: This feature applies to SAS 9.4M5 and later releases.

If a BY statement is in effect for the data, then you can uniquely name image files based on BY groups. You do this by inserting any of the #BY* substitution items into the image filename. You can specify BY lines, BY variable values, or BY variable names for IMAGENAME=. You can combine the #BY* substitution item with other text. To do this, insert the #BY* item in the specified text string at the position where you want the substitution text to appear.

As explained in the IMAGENAME= description, the file extension is automatically generated based on the OUTPUTFMT= option.
Here are descriptions of the #BY* substitution items:

**#BYLINE**

substitutes the entire BY line without leading or trailing blanks for #BYLINE in the filename string. The BY line uses the format `variable-name=value`. However, the “=” character in the BY line is replaced with an underscore character.

**Examples**
The following example specifies the BY line as the image filename:

```
imagename=#byline
```

The following example specifies the BY line preceded by the text `sales_`:

```
imagename=sales_#byline
```

**#BYVALn | #BYVAL(BY-variable-name)**

substitutes the current value of the specified BY variable for #BYVAL in the filename string. Specify the variable with one of these:

- `n` specifies a variable by its position in the BY statement. You must provide the position even if your BY statement contains only one variable. Otherwise, no substitution occurs.
- `BY-variable-name` specifies a variable from the BY statement by its name. `BY-variable-name` is not case sensitive.

**Examples**
The following example specifies the first variable in the BY statement. The value of that variable is used for the image filename.

```
imagename=#byval1
```

The following example specifies the second variable in the BY statement preceded by the text `sales_`:

```
imagename=sales_#byval2
```

The following example specifies the YEAR variable in the BY statement. The value of that variable is used for the image filename:

```
imagename=#byval(year)
```

The following example specifies the YEAR variable in the BY statement followed with the text `_sales`:

```
imagename=#byval(year)_sales
```

**#BYVARn | #BYVAR(BY-variable-name)**

substitutes the name of the BY variable or the label associated with the variable (whatever the BY line would normally display) for #BYVAR in the filename string. Specify the variable with one of these:

- `n` specifies a variable by its position in the BY statement. You must provide the position even if your BY statement contains only one variable. Otherwise, no substitution occurs.
- `BY-variable-name` specifies a variable from the BY statement by its name. `BY-variable-name` is not case sensitive.

**Examples**
The following example specifies the first variable in the BY statement. The name of that variable is used for the image filename.
The following example specifies the second variable in the BY statement preceded by the text `sales_`:

```
imagename="sales_#byvar2"
```

The following example specifies the YEAR variable in the BY statement. The name of that variable is used for the image filename:

```
imagename="#byvar(year)"
```

The following example specifies the YEAR variable in the BY statement followed with the text `_sales`:

```
imagename="#byvar(year)_sales"
```

**Note:** When using this feature, consider any filename length restrictions that might apply to your platform host. The filename length can vary greatly depending on the input data.

---

### Specifying the Image Format

Each ODS destination uses a default format for its output. You can use the OUTPUTFMT= option in the ODS GRAPHICS statement to change the output format.

**Note:** Unless you have a special requirement for changing the image format, we recommend that you not change it. The default PNG or vector graphic format is far superior to other formats, such as GIF, in support for transparency and a large number of colors. Also, PNG and vector graphics images require much less disk storage space than JPEG or TIFF formats.

If you want to generate vector graphics images, you can use the following OUTPUTFMT= values for each destination:

<table>
<thead>
<tr>
<th>ODS Destination</th>
<th>OUTPUTFMT=value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS EPUB</td>
<td>OUTPUTFMT=SVG</td>
</tr>
<tr>
<td>ODS destination for Excel</td>
<td>OUTPUTFMT=EMF</td>
</tr>
<tr>
<td>ODS HTML</td>
<td>OUTPUTFMT=SVG</td>
</tr>
<tr>
<td>ODS LISTING</td>
<td>OUTPUTFMT=EMF</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=PDF</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=PS</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=SVG</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=PCL</td>
</tr>
</tbody>
</table>
### ODS Destination

<table>
<thead>
<tr>
<th>ODS Destination</th>
<th>OUTPUTFMT=value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS PDF</td>
<td>Vector graphics images are generated by default</td>
</tr>
<tr>
<td>ODS PCL</td>
<td>OUTPUTFMT=PCL (for PCL output)</td>
</tr>
<tr>
<td>ODS PS</td>
<td>OUTPUTFMT=PS (for PostScript output)</td>
</tr>
<tr>
<td>ODS destination for PowerPoint</td>
<td>OUTPUTFMT=EMF</td>
</tr>
<tr>
<td>ODS PRINTER</td>
<td>OUTPUTFMT=PCL (for PCL output)</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=PDF (for PDF output)</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=PS</td>
</tr>
<tr>
<td></td>
<td>OUTPUTFMT=SVG</td>
</tr>
<tr>
<td>ODS RTF</td>
<td>OUTPUTFMT=EMF</td>
</tr>
<tr>
<td>ODS Measured RTF</td>
<td>OUTPUTFMT=EMF</td>
</tr>
</tbody>
</table>

When a vector graphics image cannot be generated for the format that you specify, a PNG image is generated instead and is embedded in the specified output file. The output file format and extension are not changed in that case. In the following cases, a vector graphics image cannot be generated:

- surface plots
- bivariate histograms
- graphs that use smooth gradient contours
- graphs that include continuous legends
- graphs that use data skins
- graphs that use transparency (EMF and PS ODS destinations only)
- graphs that contain one or more rotated images
- graphs that have a broken axis
- graphs that contain outline marker characters

Starting with **SAS 9.4M2**, additional cases for which vector graphics output cannot be generated for graphs are as follows:

- graphs that use gradient fill for bars in a bar chart, histogram, or waterfall chart
- graphs that use the back-light effect on text
- graphs that include a text plot that displays text with an outlined bounding box or text with a filled bounding-box background
- graphs that include images (PostScript output only)

Starting with **SAS 9.4M3**, vector graphics output can be generated in the EMF, PDF, and SVG output formats for the following cases:

- graphs that use data skins
Note: For the EMF, PDF, and SVG formats, vector graphics output is not supported for graphs that use transparency and data skins. An image is generated in that case.

- graphs that include one or more rotated images
- graphs that use gradient fills (except PDF)
- graphs that use a continuous legend

Note: For the PDF output format, vector graphics output is not supported for graphs that use a continuous legend and data transparency. An image is generated in that case.

Starting with SAS 9.4M5, vector graphics output can or cannot be generated in the output formats as indicated in the following table:

Table 4.5 Support for Vector Graphics in SAS 9.4M5

<table>
<thead>
<tr>
<th>Case</th>
<th>EMF</th>
<th>PCL</th>
<th>PDF</th>
<th>PS</th>
<th>SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surface plots</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Gradient fill contour plots</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3-D bivariate histograms</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Continuous legends without data or fill transparency</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Continuous legends with data transparency</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data skins</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data skins with transparency</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data transparency, without images</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Transparent images</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Images with data transparency</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Rotated images</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Text plots with backlight</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Text plots with fill and outline</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Scatter plots with an outlined marker character</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Broken axis</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fill patterns</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Case | EMF | PCL | PDF | PS | SVG |
--- | --- | --- | --- | --- | --- |
Fill pattern legend chiclets | No | No | No | No | No |

Note: The SGMAP procedure, which is new in SAS 9.4M5, does not support vector-based output.

Supported File Types for Output Destinations

The following table lists all of the supported file types for some ODS output destinations.

**Table 4.6 Supported File Types for Output Destinations**

<table>
<thead>
<tr>
<th>Output Destination</th>
<th>Supported File Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPUB, EPUB2</td>
<td>PNG (default), GIF, JPG, SVG</td>
</tr>
<tr>
<td>EPUB3</td>
<td>SVG (default), PNG, GIF, JPG</td>
</tr>
<tr>
<td></td>
<td>Note: EPUB3 was added in SAS 9.4M1.</td>
</tr>
<tr>
<td></td>
<td>Note: Starting with SAS 9.4M3, EPUB3 is an alias for EPUB, and the EPUB3 supported file types supersede the EPUB supported file types.</td>
</tr>
<tr>
<td>ODS destination for Excel</td>
<td>PNG (default), JPEG, JPG, EMF</td>
</tr>
<tr>
<td>HTML</td>
<td>PNG (default), GIF, JPEG, JPG, SVG</td>
</tr>
<tr>
<td>HTML5</td>
<td>SVG (default), PNG, GIF, JPEG, JPG</td>
</tr>
<tr>
<td>LISTING</td>
<td>PNG (default), BMP, EMF, EPSI, GIF, JFIF, JPEG, JPG, PDF, PS, SASEMF, STATIC, TIFF, WMF, PSL, SVG</td>
</tr>
<tr>
<td>PDF</td>
<td>Native PDF (default), JPEG, JPG, GIF, PNG</td>
</tr>
<tr>
<td>ODS destination for PowerPoint</td>
<td>PNG (default), JPEG, JPG, GIF, EMF, TIFF, BMP</td>
</tr>
<tr>
<td>PS</td>
<td>PNG (default), JPEG, JPG, GIF, EPS, PDF, PCL, PS</td>
</tr>
<tr>
<td>RTF and Measured RTF</td>
<td>EMF (default), PNG, JPEG, JPG, JFIF</td>
</tr>
</tbody>
</table>

Description of Supported File Types

The following table provides descriptions of the supported file types for ODS output destinations.
<table>
<thead>
<tr>
<th>File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMP (Microsoft Windows Device Independent Bitmap)</td>
<td>Supports color-mapped and true color images that are stored as uncompressed or run-length encoded data. BMP was developed by Microsoft Corporation.</td>
</tr>
<tr>
<td>CGM (Computer Graphics Metafile)</td>
<td>A free and open international standard file format for 2-D vector graphics, raster graphics, and text. This format is defined by ISO/IEC 8632.</td>
</tr>
<tr>
<td>DIB (Microsoft Windows Device Independent Bitmap)</td>
<td>See the description of BMP.</td>
</tr>
<tr>
<td>EMF Plus (Enhanced Metafile Format Plus Extensions)</td>
<td>Supports Enhanced Metafile Plus Extensions that provides additional functionality, such as support of RGBA colors.</td>
</tr>
<tr>
<td>EMF Dual (Enhanced Metafile Format and Enhanced Metafile Format Plus Extensions)</td>
<td>Produces both EMF and EMF Plus formats simultaneously in the same output.</td>
</tr>
<tr>
<td>EPS</td>
<td>Encapsulated PostScript</td>
</tr>
<tr>
<td>EPSI (Microsoft NT Enhanced Metafile)</td>
<td>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.</td>
</tr>
<tr>
<td>GIF (Graphics Interchange Format)</td>
<td>Supports only color-mapped images. GIF is owned by CompuServe, Inc.</td>
</tr>
<tr>
<td>JFIF (JPEG File Interchange Format)</td>
<td>Supports JPEG image compression. JFIF software is developed by the Independent JPEG Group.</td>
</tr>
<tr>
<td>JPEG or JPG (Joint Photographic Experts Group)</td>
<td>A file format that is used for storing noninteractive images.</td>
</tr>
<tr>
<td>PBM (Portable Bitmap Utilities)</td>
<td>Supports gray-scale, color, RGB, and bitmap files. The Portable Bitmap Utilities are a set of free utility programs that were developed primarily by Jef Poskanzer.</td>
</tr>
<tr>
<td>File Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>PCL</td>
<td>Printer Control Language</td>
</tr>
<tr>
<td>PNG (Portable Network Graphic)</td>
<td>Supports true color, gray-scale, and 8-bit images.</td>
</tr>
<tr>
<td>PS (PostScript Image File Format)</td>
<td>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.</td>
</tr>
<tr>
<td>PSL (PostScript)</td>
<td>PostScript</td>
</tr>
<tr>
<td>STATIC</td>
<td>Chooses the best image format for the current ODS destination.</td>
</tr>
<tr>
<td>SVG (Scalable Vector Graphics)</td>
<td>Is an XML language for describing two-dimensional vector graphics.</td>
</tr>
<tr>
<td>TIFF (Tagged Image File Format)</td>
<td>Internally supports a number of compression types and image types, including bitmapped, color-mapped, grayscale, and true color. TIFF was developed by Aldus Corporation and Microsoft Corporation and is used by a wide variety of applications (available if licensed).</td>
</tr>
<tr>
<td>XBM</td>
<td>X Window Bitmap</td>
</tr>
<tr>
<td>XPM</td>
<td>X Window Pixmap</td>
</tr>
</tbody>
</table>

Temporarily Saving and Restoring ODS GRAPHICS Settings

About PUSH and POP

Note: This feature applies to SAS 9.4M3 and later releases.

Although you can use the RESET option to restore the default ODS GRAPHICS settings, there might be times when you want to save your current custom settings and later restore them. ODS enables you to temporarily store your custom settings in a stack created for this purpose, perform some other task with different settings, and then restore the previous settings.

The PUSH option saves the current ODS GRAPHICS settings to the stack and increments the stack depth. The POP option restores the most recently stored settings from the stack and decrements the stack depth.
This feature is useful when you write macros. Within a macro you can PUSH at the start of the macro and POP at the end. This enables your macro to have custom ODS GRAPHICS behaviors without affecting the calling environment.

You can specify PUSH as many times as you like up to the limit that is defined by the STACKDEPTHEXMAX= option. The pushed settings remain in the stack in the current SAS session until they are popped or the stack is emptied. For more information, see “Managing the Stack Depth” on page 68. For each PUSH option, you can specify a POP option. ODS issues a warning if you specify POP without a corresponding PUSH. In that case, nothing is popped because nothing has been pushed to the stack.

Order of specification is important when using the PUSH option. For example, the following statement pushes the NOBORDER option to the stack along with any other custom settings that are in effect.

```sas
ods graphics / noborder push;
```

A subsequent POP option restores the pushed settings including NOBORDER.

However, the following statement pushes the current custom settings and then sets the NOBORDER option.

```sas
ods graphics / push noborder;
```

Here, the subsequent POP option restores whatever border setting was in effect when the PUSH option was specified.

**TIP** Use the SHOW option to show the ODS GRAPHICS settings that are currently in effect.

### Settings That Can Be Pushed

The PUSH and POP options apply to all ODS GRAPHICS options except the following: PUSH, POP, RESET=INDEX, and SHOW.

### How Code Errors Affect the PUSH Operation

If the ODS GRAPHICS statement contains a syntax error, then the PUSH option is ignored.

For example, the PUSH option is ignored in the following statement:

```sas
ods graphics / antialias=bogus push;
```

A syntax error (BOGUS) in ANTIALIAS causes the parser to ignore the remaining options. However, a simple semantics error does not prevent the remaining options from being handled. In the following statement, the PUSH option is honored.

```sas
ods graphics / antialiasmax=-1 push;
```

In this statement, ANTIALIASMAX= -1 is invalid. The option expects a zero or a positive integer. In this case, a warning is issued to the log, but the PUSH occurs.

**Note:** Syntax errors in your code can have other unexpected results that are not described here.
Managing the Stack Depth

By default, the stack supports up to 1024 pushes. You can change the default by using the STACKDEPTHMAX= option.

If the specified STACKDEPTHMAX= value is less than the current stack depth, then the stack is popped until its depth equals the specified value. Popping the stack does not affect other option settings.

If you want to empty the stack, issue the following statement:

```
ods graphics / stackdepthmax=0 reset=stackdepthmax;
```

This statement first empties the stack of all PUSH requests and then restores the stack size to 1024.

ODS PREFERENCES Statement

Reverts the ODS settings back to start-up defaults.

Category: ODS: Output Control

Tip: It is useful to specify this statement if you are creating graphics and have changed your default output directory to something other than Work. When you specify the ODS PREFERENCES statement, the default directory is set to Work, which is the original default. Any output is then generated in your Work folder.

Syntax

```
ODS PREFERENCES;
```

Without Arguments

The ODS PREFERENCES statement reverts the ODS back to the default behavior:

- HTML output is created.
- LISTING output is not created.
- Both HTML and graph image files are saved in the Work folder (and not your current directory).
- Default style is HTMLBlue.
- ODS Graphics is enabled.

The changes made by the ODS PREFERENCES statement last only for the duration of your SAS session, or until you change them with an ODS statement or option. The ODS PREFERENCES statement does not change the settings in the TOOLS ➔ Options ➔ Preferences ➔ Results dialog box.

ODS RESULTS Statement

Tracks ODS output in the Results window.

Category: ODS: Output Control
Alias: ODS RESULTS|NORESULTS;
Restriction: Valid in a windowing environment only, not in batch mode.

Syntax

**ODS RESULTS ON | OFF;**
**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 TEXT= Statement**

Inserts text into your ODS output.

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 Argument

**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 quotes.

Tip The UserText style element controls text specified with the TEXT= statement.
Examples:

Example 1: Adding Text to Multiple Destinations

Features:
- ODS HTML statement
- ODS PDF statement
- ODS RTF statement
- ODS TEXT= statement

PROC TEMPLATE:
- DEFINE STYLE statement
- PARENT= statement
- STYLE statement

PROC PRINT

Details
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 that controls the font style, font color, and other attributes of the text that the ODS TEXT= statement adds.

Program

```sas
options obs=10;
proc template;
    define style Styles.MyStyle;
    parent=styles.htmlblue;
    style usertext from usertext /
        foreground=red;
    end;
run;

ods html file="text.html" style=Styles.MyStyle ;
ods pdf file="text.pdf" startpage=never notoc style=Styles.MyStyle ;
ods rtf file="text_trad.rtf" style=Styles.MyStyle ;

title "January Orders ";
footnote " For All Employees";
ods text="My Text 1";
ods text="My Text 2";
proc print data=exprev;
run;
ods text="My Text 3";
ods pdf close;
ods rtf close;
ods html close;
title;
footnote;
proc template;
    delete Styles.MyStyle ;
run;
```
Program Description

options obs=10;

**Create the Styles.MyStyle style template.** The Styles.MyStyle style template modifies the Usertext style element to change the font color of text created by the TEXT= statement to red.

```
proc template;
    define style Styles.MyStyle;
    parent=styles.htmlblue;
    style usertext from usertext /
        foreground=red;
    end;
run;
```

**Send output to multiple ODS destinations.** The following statements open the HTML, PDF, and RTF destinations. The STYLE= options specifies that the style Styles.MyStyle is applied to the output.

```
ods html file="text.html" style=Styles.MyStyle ;
ods pdf file="text.pdf" startpage=never notoc style=Styles.MyStyle ;
ods rtf file="text_trad.rtf" style=Styles.MyStyle ;
```

**Add a title and footnote.** The TITLE and FOOTNOTE statements specify the title and footnote. You must place the TITLE and FOOTNOTE statements before the ODS TEXT= statements.

```
title "January Orders ";
footnote " For All Employees";
```

**Add text strings before the output 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";
```

**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 the RTF, HTML, and PRINTER destinations and remove the titles and footnotes.** The ODS RTF CLOSE statement closes the RTF destination. The ODS PDF CLOSE statement closes the PRINTER destination. The ODS HTML CLOSE statement closes the HTML destination. The TITLE and FOOTNOTE statements remove any titles and footnotes previously specified.

```
ods pdf close;
ods rtf close;
ods html close;
title;
footnote;
```

**After your output is created, you can delete the Styles.MyStyle style template.** The DELETE statement deletes the Styles.MyStyle style template.

```
proc template;
    delete Styles.MyStyle ;
```
run;

Output

Output 4.1  HTML Output with Text Added

Output 4.2  PDF Output with Text Added
Example 2: Adding Text That Imitates a System Title

Features:
- ODS PDF statement option
  - FILE=
  - NOTOC
  - STARTPAGE=NO
  - STYLE=
- ODS NOPROCTITLE statement
- ODS HTML
- ODS TEXT statement
- PROC TEMPLATE
- OPTIONS statement
- PROC MEANS
- PROC PRINT
- TITLE statement

Details
SAS titles and footnotes are displayed once per page in the PDF destination. Therefore, when the STARTPAGE= option is set to NO (OFF) and output from more than one procedure or DATA _NULL_ step is routed to ODS PDF, only the first set of titles and footnotes are written to the output file. This example shows how to use the TEXT= option to mimic an interim title, displayed above the second procedure output. PROC TEMPLATE is used to create a custom style template that mimics the style of the Systemtitle element. Systemtitle is the style element that controls the appearance of titles.

This example requires some knowledge of style templates, style elements, and style attributes.

For complete documentation about styles, see "Understanding Styles, Style Elements, and Style Attributes" on page 432.
For a table of style attributes, see “Style Attributes Tables” on page 828.

For a table of style elements affecting pages, see “Style Elements Affecting Pages” on page 800.

For a table of style elements affecting the table of contents, see “Style Elements Affecting Tables of Contents” on page 798.

Program

```sas
ods html close;
options nodate;

proc template;
  define style styles.mimictitle;
  parent=styles.pearl;
  class usertext from systemtitle /
    just=c;
  end;
run;
ods pdf file="file.pdf" notoc startpage=no style=styles.mimictitle;

title "Overriding the Default Procedure Title";
ods noproctitle;
proc means data=sashelp.cars;
  class cylinders;
  var mpg_city mpg_highway;
run;

ods text="My Custom PROC PRINT Output Title";

proc print data=sashelp.cars noobs;
  where mpg_highway gt 45;
run;

ods pdf close;
ods html;
proc template;
delete Styles.Mimictitle;
run;
```

Program Description

**Close the HTML destination so that no HTML output is produced.** The HTML destination is open by default. The ODS HTML CLOSE statement closes the HTML destination to conserve resources. If the destination were left open, then ODS would produce both HTML and PDF output.

ods html close;

**Set the SAS system options.**

options nodate;
Create a custom style template. The Usertext style element controls the appearance of user text, which includes text specified by the TEXT= option. The Systemtitle style element controls the appearance of the default SAS titles, which include the text specified by the TITLE= statement. This PROC TEMPLATE step creates a new style named Styles.Mimictitle, which contains all of the style elements and style attributes that Styles.Pearl contains. However, the CLASS statement modifies the Usertext style element to produce center-justified text, just as the Systemtitle style element does. This ensures that text specified by the TEXT= option will look the same as the text specified by the TITLE= statement.

```sas
proc template;
  define style styles.mimictitle;
  parent=styles.pearl;
  class usertext from systemtitle / just=c;
end;
run;
```

Open the PDF destination and specify the ODS PDF statement options. The ODS PDF statement opens the PDF destination and the FILE= option specifies the PDF filename. The NOTOC option specifies that no table of contents is created. The STARTPAGE=NO option specifies that no new pages are inserted at the beginning of each procedure, or within certain procedures, even if new pages are requested by the procedure code. The STYLE= option specifies the style to use for the output, which is Styles.Mimictitle in this example.

```sas
ods pdf file="file.pdf" notoc startpage=no style=styles.mimictitle;
```

Specify a title.

```sas
title "Overriding the Default Procedure Title";
```

Create the MEANS procedure output and suppress the procedure title. The ODS NOPROCTITLE statement suppresses the writing of the procedure title that produces the results.

```sas
ods noproctitle;
proc means data=sashelp.cars;
  class cylinders;
  var mpg_city mpg_highway;
run;
```

Specify the text to use as the second procedure title. The TEXT= option specifies the text that appears above the PRINT procedure output. Because the custom style Styles.Mimictitle was specified in the ODS PDF statement, this text will look like a title specified by the TITLE= statement.

```sas
ods text="My Custom PROC PRINT Output Title";
```

Create the PRINT procedure output.

```sas
proc print data=sashelp.cars noobs;
  where mpg_highway gt 45;
run;
```

Close the PDF destination and open the HTML destination. The ODS PDF CLOSE statement closes 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. The ODS HTML statement opens the HTML destination and returns SAS to the default ODS destination.

```sas
ods pdf close;
ods html;
```

**After your output is produced, you can remove the custom style.** The DELETE statement in PROC TEMPLATE removes the custom style.

```sas
proc template;
    delete Styles.Mimictitle;
run;
```

**PDF Output**
The following image shows the text “My Custom PROC PRINT Output Title” above the PROC PRINT table in the same style as the title.

**Output 4.4  PDF Output with Custom Procedure Title**

---

**ODS TRACE Statement**

Writes to the SAS log a record of each output object that is created, or suppresses the writing of this record.

**Category:** ODS: Output Control

**Default:** OFF

Syntax

```
ODS TRACE ON <options> ;  
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  YES

Optional Arguments

```
DOM<="external-file">  
```

specifies that the ODS document object model is written to the SAS log or an external file.

```
external-file  
```

is the name of an external output file.

 Requirements  You must enclose external-file in quotation marks.

See  For complete documentation about the ODS document object model, see "Working with the ODS Document Object Model" in SAS Output Delivery System: Advanced Topics.

```
EXCLUDED
```

includes, in the trace record, information for excluded output objects.

 Example  “Example 2: Conditionally Selecting Output Objects” on page 32

```
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 template. The trace record provides information about the data component, the table template, 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 about how to reference an output object, see “How ODS Determines the Destinations for an Output Object” in SAS Output Delivery System: User’s Guide. For example, you could use this name in an ODS OUTPUT statement to make a data set from the output object. You could also use this name 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 template that ODS uses to format the output object. You can modify this definition with PROC TEMPLATE. See the “EDIT Statement” on page 536 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. You could also use the path 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 option LABEL on page 78.

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, the following is the full path of the output object:
  Univariate.City.Pop_90.TestsForLocation

- 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, suppose the full path is the following:
Then the partial paths are as follows:

- a label that is enclosed in quotation marks. For example:
  "The UNIVARIATE Procedure"

- a label path. For example, the following is the label path for the output object:
  "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, suppose the label path is the following:
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"

Then the partial label paths are as follows:

- "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: Determining Which Output Objects a Procedure Creates

Features: ODS TRACE statement:
  LABEL
  OFF
  ON
  PROC UNIVARIATE

Details
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.

Program

ods trace on / label;
proc univariate data=statepop mu0=3.5;
  var citypop_90 citypop_80;
run;
ods trace off;
**Program Description**

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;
```

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"
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.
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.CityPop_90.Moments
Label Path: "The Univariate Procedure"."CityPop_90"."Moments"

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"

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"

Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.CityPop_90.Quantiles
Label Path: "The Univariate Procedure"."CityPop_90"."Quantiles"

Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.CityPop_90.ExtremeObs
Label Path: "The Univariate Procedure"."CityPop_90"."Extreme Observations"
See Also

Statements

- "ODS EXCLUDE Statement" on page 14
- "ODS SELECT Statement" on page 22

ODS PROCLABEL Statement

Enables you to change a procedure label.

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.
Examples:
“Example 2: Adding Text That Imitates a System Title” on page 73
“Customizing the Table of Contents” in SAS Output Delivery System: User’s Guide

Syntax

**ODS PROCLABEL** `string`;
**ODS PROCLABEL=** `string`;

Required Argument

`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
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.

Category: ODS: Output Control

Interaction: The combination of the ODS DOCUMENT statement and the DOCUMENT procedure enables you to store a report’s individual components and then modify and replay the report. The ODS DOCUMENT statement stores the actual ODS objects that are created when running a report. You can then use the DOCUMENT procedure to rearrange, duplicate, or remove output from the results of a procedure or a database query without invoking the procedures from the original report. For complete documentation about the DOCUMENT procedure, see Chapter 6, “DOCUMENT Procedure,” on page 93.

Syntax

**ODS DOCUMENT** *action*;

**ODS DOCUMENT**

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

< **DIR**=<( **PATH**=path<(access–option)> ) < **LABEL**="label" > )>

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

Actions

The following actions are available for the ODS DOCUMENT statement:

**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.

<table>
<thead>
<tr>
<th>Default</th>
<th>NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>The DOCUMENT destination must be open for this action to take effect.</td>
</tr>
<tr>
<td>See</td>
<td>“ODS EXCLUDE Statement” on page 14</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Default</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>The DOCUMENT destination must be open for this action to take effect.</td>
</tr>
<tr>
<td>See</td>
<td>“ODS SELECT Statement” on page 22</td>
</tr>
</tbody>
</table>

**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. |

Optional Arguments

**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 following form.

path#$sequence–number

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.

Interaction If a label is specified with the LABEL= option, then overrides any existing label assigned to the document.

Tip If the ODS document does not exist, then it is created.

CAUTION If the ODS document already exists, then it is overwritten.

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

Interaction If a label is specified with the LABEL= option, then it is assigned to the document.

Tip If the ODS document does not exist, then the document is created.
CAUTION If the document already exists, then its contents are not changed.

Default UPDATE

Note Procedure output or data queries are 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.

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

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.

Interaction If a label is specified with the LABEL= option, then it overrides any existing label assigned to the document.

Tip If the ODS document does not exist, then it is created.

CAUTION If the ODS document already exists, then it is overwritten.

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

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

Tip If the ODS document does not exist, then the document is created.

CAUTION If the document already exists, then its contents is not changed.

Default UPDATE

Interaction When a DOCUMENT destination is open, using the NAME= option in an ODS DOCUMENT statement forces ODS to close the destination and all files associated with it, and to open a new instance of the destination.
# DOCUMENT Procedure

## Overview: DOCUMENT Procedure

- What Does the DOCUMENT Procedure Do?  
- DOCUMENT Procedure Terminology

## Concepts: DOCUMENT Procedure

- Understanding an ODS Document Path
- Understanding Sequence Numbers
- ODS Documents and Base SAS Procedures
- Getting Familiar with Output Objects
- Compatibility across SAS Versions

## Syntax: DOCUMENT Procedure

- PROC DOCUMENT Statement
- COPY TO Statement
- DELETE Statement
- DIR Statement
- DOC Statement
- DOC CLOSE Statement
- HIDE Statement
- IMPORT TO Statement
- LINK Statement
- LIST Statement
- MAKE Statement
- MOVE TO Statement
- NOTE Statement
- OBANOTE Statement
- OBBNOTE Statement
- OBDYNAM Statement
- OBFOOTN Statement
- OBPAGE Statement
- OBSTITLE Statement
- OBTEMPL Statement
- OBTITLE Statement
- RENAME TO Statement
- REPLAY Statement
- SETLABEL Statement
- UNHIDE Statement

## Usage: DOCUMENT Procedure

- Working with ODS Documents
- Replaying Graphics
- Customizing Labels, Titles, and Footnotes with BY Variables
- Using the Results Window in the SAS Windowing Environment
- Using the Documents Window in the SAS Windowing Environment
- Comparisons between the Documents Window and the Results Window in the SAS Windowing Environment
Overview: DOCUMENT Procedure

What Does the DOCUMENT Procedure Do?

The combination of the ODS DOCUMENT statement and the DOCUMENT procedure enables you to store a report’s individual components and then modify and replay the report. The ODS DOCUMENT statement stores the actual ODS objects (or references to those objects) that are created when running a report. You can then use the DOCUMENT procedure to rearrange, duplicate, or remove output from the results of a procedure or a database query without invoking the procedures from the original report. You can also use the DOCUMENT procedure to do the following:

- transform a report without rerunning an analysis or repeating a database query
- modify the structure of output
- display output to any ODS output format
- navigate the current directory and list entries
- open and list ODS documents
- manage output

With the DOCUMENT procedure, you are not limited to simply regenerating the same report. You can change the order in which objects are rendered, the table of contents, the templates that are used, macro variables, and ODS system options.

Unlike other ODS destinations, in the SAS Windowing Environment the DOCUMENT destination has a graphical user interface (GUI), called the Documents window, 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 DOCUMENT Procedure ” on page 158. For an example of using the Documents window to rearrange output, see “Restructuring Output” in SAS Output Delivery System: Advanced Topics. For an example of replaying output with the Documents window, see “Replaying a Document Using the Document Window” on page 157.
DOCUMENT Procedure Terminology

- **current document**
  - is the open document.

- **current directory**
  - is your current location in the open document. The "^" symbol represents the current directory.

- **entry**
  - is a directory, output object, note, or link.

- **graph segment**
  - is an output object that contains a graph.

- **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 an ODS document, leading to a particular entry. The "^" symbol represents the current directory and the "^^" symbol represents the parent directory.

- **replay**
  - is the regeneration of output, in the same or different format, without rerunning analyses or data queries.

- **root directory**
  - is the top level of an ODS document. The root is not contained within another directory and it does not have a name assigned. The root is similar to the root directory of a Windows file system.

Concepts: DOCUMENT Procedure

Understanding an ODS Document Path

**Definition of ODS Document Path**

An ODS document is stored as an item store. This file format enables client applications to define a hierarchical file system within a file. This is similar to a directory system in a UNIX or Windows operating environment. Therefore, an ODS document path indicates the location of an entry. In the preceding output, the document path for the entry **Country=Canada** is `\Tabulate#1\ByGroup1#1`.

**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 that is specified in the operating environment
- can have labels that are no more than 256 characters

Entries in an ODS document can be displayed 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 directory, 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 directory. Entries are assigned sequence numbers according to the sequence in which they are added to a directory. 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 have an initial number of 1.

### ODS Documents and Base SAS Procedures

You can create an ODS document from almost any SAS procedure. The PRINT, REPORT, and TABULATE procedures use table templates that are created by the user and not defined by an external 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 PRINT, TABULATE, and REPORT procedures.

### 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 the following 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 are displayed 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 are displayed before the output object.

footnote
is created by the FOOTNOTE statement and is displayed at the bottom of the page.

page break
causes a page break before 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 at the top of the page.

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

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.
Syntax: DOCUMENT Procedure

PROC DOCUMENT <options>;
    COPY path <(where-expression)> <, path-2 <(where-expression-2)>, ...> TO path </options>;
    DELETE path <(where-expression)> <, path-2 <(where-expression-2)>, ...> < / LEVELS= ALL | value>;
    DIR <path>;
    DOC <options>;
    DOC CLOSE;
    HIDE path <, path-2, ...>;
    IMPORT DATA= data-set-name<data-set-options> | GRSEG=grseg | TEXTFILE=filename | fileref TO path </options>;
    LINK path TO path </options>;
    LIST path <(where-expression)> <, path-2 <(where-expression-2)>, ...> </options>;
    MAKE path <, path-2, ...> </options>;
    MOVE path <(where-expression)> <, path-2 <(where-expression-2)>, ...> TO path </options>;
    NOTE path <"text"> </options>;
    OBANOTE<n> output-object <"text"> </option>;
    OBBNOTE<n> output-object <"text"> </option>;
    OBDYNAM output-object;
    OBFOOTN<n> output-object <"text">;
    OBPAGE output-object </options>;
    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)>, ...> </options>;
    SETLABEL path 'label';
    UNHIDE path <, path-2, ...>;
QUIT;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC DOCUMENT</td>
<td>Render ODS output without rerunning procedures and gain more control over the structure and hierarchy of the output</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6</td>
</tr>
<tr>
<td>COPY TO</td>
<td>Insert a copy of an entry into a specified path</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete entries from a specified path or paths</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Task</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>DIR</td>
<td>Set or display the current directory</td>
<td>Ex. 1, Ex. 2, Ex. 5</td>
</tr>
<tr>
<td>DOC</td>
<td>Open a document and its contents to browse or edit</td>
<td>Ex. 1, Ex. 5</td>
</tr>
<tr>
<td>DOC CLOSE</td>
<td>Close the current document</td>
<td></td>
</tr>
<tr>
<td>HIDE</td>
<td>Prevent output from being displayed when the document is replayed</td>
<td></td>
</tr>
<tr>
<td>IMPORT TO</td>
<td>Import a data set or graph segment into the current directory</td>
<td>Ex. 4</td>
</tr>
<tr>
<td>LINK</td>
<td>Create a symbolic link from one output object to another output object</td>
<td></td>
</tr>
<tr>
<td>LIST</td>
<td>List the content of one or more entries</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6</td>
</tr>
<tr>
<td>MAKE</td>
<td>Create one or more new directories</td>
<td></td>
</tr>
<tr>
<td>MOVE TO</td>
<td>Move entries from one directory to another directory</td>
<td></td>
</tr>
<tr>
<td>NOTE</td>
<td>Create text strings in the current directory</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>OBANOTE</td>
<td>Create or modify lines of text after the specified output object</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>OBBNOTE</td>
<td>Create or modify lines of text before the specified output object</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>OBDYNAM</td>
<td>Print a table of dynamic values for the specified output object</td>
<td>Ex. 5, Ex. 6</td>
</tr>
<tr>
<td>OBFOOTN</td>
<td>Create or modify lines of text at the bottom of the page in which the output object is displayed</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>OBPAGE</td>
<td>Create or delete a page break for an output object</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>OBSTITLE</td>
<td>Create or modify subtitles</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>OBTEMPL</td>
<td>Write the source code of the ODS template that is associated with a specified output object</td>
<td>Ex. 3</td>
</tr>
<tr>
<td>OBTITLE</td>
<td>Create or modify lines of text at the top of the page where the output object is displayed</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>RENAME TO</td>
<td>Assign a different name to a directory or output object</td>
<td></td>
</tr>
<tr>
<td>REPLAY</td>
<td>Replay one or more entries to the specified open ODS destinations</td>
<td>Ex. 1, Ex. 2, Ex. 4, Ex. 6</td>
</tr>
</tbody>
</table>
PROC DOCUMENT Statement

Creates or opens a document to modify.

Default: Documents are opened in the UPDATE access mode.

Restriction: User-defined format names must be unique within an ODS DOCUMENT.

Note: If the DOCUMENT destination is not closed with an ODS DOCUMENT CLOSE statement, then ODS continues to append files to the document.

Syntax

PROC DOCUMENT <options <access-options>>;

Optional Arguments

NAME= <libref.>member-name <access-options>

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-options

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

is an access-option that 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

is an access-option that opens a document and provides Read and Write access. For example, the following PROC DOCUMENT statement opens the document Work.YourDoc in Write mode:

```sas
proc document name=yourdoc(write);
run;
```

**Interaction**

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

**Tip**

If the ODS document does not exist, then it is created.

**CAUTION**

If the ODS document already exists, then it is overwritten.

UPDATE

is an access-option that opens an ODS document and appends new content to the document. UPDATE provides Update access as well as Read access.

**Interaction**

If a label has been specified with the LABEL= option, then it is assigned to the document.

**Tip**

If the ODS document does not exist, then the document is created.

**CAUTION**

If the document already exists, then its contents is not changed.

**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:

```sas
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.

**Example:**

“Using PROC DOCUMENT to Work with Table Templates ” in SAS Output Delivery System: Advanced Topics

**Syntax**

```
COPY path <(where-expression)> <, path-2 <(where-expression-2)>, ...> TO path <\ options>;
```
Required Argument

_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 directory and the "^^" symbol represents the parent directory.

Optional Arguments

**AFTER= path**

inserts a copy of an entry after the specified path.

Tip You can use the symbol "^" to represent the current directory and the symbol "^^" to represent the parent directory.

**BEFORE= path**

inserts a copy of an entry before the specified path.

Tip You can use the symbol "^" to represent the current directory and the symbol "^^" to represent the parent directory.

**FIRST**

inserts a copy of an entry at the beginning of the specified directory. For example, the following COPY TO statement inserts a copy of the entry Monday_Report at the beginning of the root directory:

```plaintext
copy weekly\monday_report to \ / first;
run;
```

**LAST**

inserts a copy of an entry at the end of the specified directory.

**LEVELS= ALL | value**

specifies the number of levels that you want to copy.

ALL specifies all levels.

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:

```plaintext
copy weekly to \work.mydoc\monthly / levels = 2;
run;
```

Default ALL

Restriction The LEVELS= option is valid only when you specify a directory.

**(WHERE=(where-expression-1<operator>))**

conditionally selects a subset of entries in an ODS document.

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 value.

**SAS function**

For information about SAS functions, see *SAS Functions and CALL Routines: Reference*.

**subsetting variable**

is a special type 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:

```
moves (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, suppose that this is the ODS TRACE label path:
Note that in document label paths, the instances of '.' are replaced with '\'.

See "ODS TRACE Statement" on page 76

Example  The following LIST statement lists all items containing "Fit Statistics" in the label path.

\lstinline|list glm (where=(_labelpath_ ? "Fit Statistics")\)/ levels=all;
run;|

_\textbf{MAX}_

is the last observation.

Restrictions  _\textbf{MAX}_ is used only for output objects.

Example  The following REPLAY statement replays all observations except the last observation:

\lstinline|replay class(where=(_obs_ < _max_));|

_\textbf{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:

\lstinline|move ^(where=(_type_ = 'Graph' and _mdate_ = '16JUL2004'd)) to \work.mydoc\monthly; run;|

_\textbf{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:

\lstinline|replay ^(where=(_mdatetime_ = '01may04:9:30:00'dt)); run;|

_\textbf{MIN}_

is the first observation.

Restrictions  _\textbf{MIN}_ is used only for output objects.

Example  The following REPLAY statement replays all observations except the first observation:

\lstinline|replay class(where=(_obs_ < _min_));|

_\textbf{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:

\lstinline|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:

del glm(where=_name_ ? 'stemleng');

_OBS_

is the current observation number in an output object.

Restrictions

_OBS_ is used only for output objects.

_OBS_ is used only in the REPLAY statement.

Examples

The following REPLAY statement replays all but the first ten observations:

replay class(where=_obs_ > 10));

The following REPLAY statement replays all observations except the last observation:

replay class(where=_obs_ < _max_));

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observations:

replay class(where=_obs_ in (1,3,5,7,9));

_observation-number

is the observation number to be replayed.

Restrictions

_observation-number_ is used only for output objects.

_observation-number_ is used only in the REPLAY statement.

Example

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observation:

replay class(where=_obs_ in (1,3,5,7,9));

_observation-variable

is the name of an observation.

Restrictions

_observation-variable_ is used only for output objects.

_observation-variable_ is used only in the REPLAY statement.

Examples

The following REPLAY statement replays all observations where the variable Weight is greater than 100:

replay class(where={weight>100});

The following REPLAY statement replays all observations where the variable Sex is equal to 'F'.

replay class(where={sex='F'});

葩_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.

See

"Understanding Sequence Numbers " on page 96

Example

The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:
replay glm(where=(_seqno_ = 2));

_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:
motion ^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:
motion ^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 6.1 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or =~ or ` or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>Symbol</td>
<td>Mnemonic Equivalent</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------</td>
<td>------------</td>
</tr>
<tr>
<td>IN</td>
<td>Equal to one from a list of values</td>
<td></td>
</tr>
</tbody>
</table>

**Restriction**

For the REPLAY statement, the WHERE= option applies to directories and output objects. For the following statements, the WHERE= option applies to directories only:

- COPY TO
- DELETE
- LIST
- MOVE TO

**Requirement**

Enclose where-expression in quotation marks.

**See**

For advanced information about using where-expressions, see “Using WHERE Expressions in PROC DOCUMENT” in SAS Output Delivery System: Advanced Topics.

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**

“Opening and Listing ODS Documents Using WHERE Expressions” in SAS Output Delivery System: Advanced Topics

---

**DELETE Statement**

Deletes entries from the current directory.

**Restriction:**

The root directory cannot be deleted or moved.

**Note:**

The DELETE statement affects all levels of a directory below the specified path.

**Syntax**

```
DELETE path *(where-expression)* path-2 *(where-expression-2)*,…>
</LEVELS=ALL | value>
```

**Required Argument**

`path`

specifies the location of one or more links, output objects, or directories. 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 directory and the symbol `^^` to represent the parent directory.

Optional Arguments

**LEVELS= ALL | value**

- **ALL** specifies all levels.
- **value** specifies the numeric value of the path level.

Default **ALL**

Restriction The **LEVELS=** option is valid only when you specify a directory.

**WHERE=(where-expression-1<operator>))**

conditionally selects a subset of entries in an ODS document.

- **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 value.
  - **SAS function** For information about SAS functions, see *SAS Functions and CALL Routines: Reference*.
  - **subsetting variable** is a special type 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.
    - **_CDATETIME_** is the creation datetime 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;
```

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'));
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, suppose that this is the ODS TRACE label path:
Note that in document label paths, the instances of '.' are replaced with '\'.

See “ODS TRACE Statement” on page 76

Example The following LIST statement lists all items containing “Fit Statistics” in the label path.
list glm (where=(_labelpath_ ? "Fit Statistics")/ levels=all);
run;

_max_

is the last observation.

Restrictions _max_ is used only for output objects.

_max_ is used only in the REPLAY statement.

Example The following REPLAY statement replays all observations except the last observation:
replay class(where=(_obs_ < _max_));

_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')) to \work.mydoc\monthly;
run;
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;
```

is the first observation.

Restrictions

_MIN_ is used only for output objects.

_MIN_ is always set to 1

_MIN_ is used only in the REPLAY statement.

Example

The following REPLAY statement replays all observations except the first observation:

```
replay class(where=(obs_ < _min_));
```

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;
```

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'));
```

is the current observation number in an output object.

Restrictions

_OBS_ is used only for output objects.

_OBS_ is used only in the REPLAY statement.

Examples

The following REPLAY statement replays all but the first ten observations:

```
replay class(where=(obs_ > 10));
```

The following REPLAY statement replays all observations except the last observation:

```
replay class(where=(obs_ < _max_));
```

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observations:

```
replay class(where=(obs_ in {1,3,5,7,9}));
```

observation-number is the observation number to be replayed.
Restrictions  *observation-number* is used only for output objects.

*observation-number* is used only in the REPLAY statement.

Example  The following REPLAY statement replays the first, third, fifth, seventh, and ninth observation:

```plaintext
replay class(where=(_obs_ in (1,3,5,7,9)));
```

*observation-variable* is the name of an observation.

Restrictions  *observation-variable* is used only for output objects.

*observation-variable* is used only in the REPLAY statement.

Examples  The following REPLAY statement replays all observations where the variable Weight is greater than 100:

```plaintext
replay class(where=(weight>100));
```

The following REPLAY statement replays all observations where the variable Sex is equal to 'F'.

```plaintext
replay class(where=(sex='F'));
```

_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:

```plaintext
list ^(where=(_path_ ? 'Anova'));
run;
```

_SEQNO_  is the sequence number of the current entry.

See "Understanding Sequence Numbers" on page 96

Example  The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:

```plaintext
replay glm(where=(_seqno_ = 2));
```

_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:

```plaintext
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**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**Restriction**

For the REPLAY statement, the WHERE= option applies to directories and output objects. For the following statements, the WHERE= option applies to directories only:
- COPY TO
- DELETE
- LIST
- MOVE TO

**Requirement**

Enclose where-expression in quotation marks.

**See**

For advanced information about using where-expressions, see "Using WHERE Expressions in PROC DOCUMENT" in SAS Output Delivery System: Advanced Topics.

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**

“Opening and Listing ODS Documents Using WHERE Expressions” in SAS Output Delivery System: Advanced Topics
DIR Statement
Sets or displays the current directory.

Examples:
- “Example 1: Navigating the Directory and Listing the Entries” on page 162
- “Opening and Listing ODS Documents Using WHERE Expressions” in SAS Output Delivery System: Advanced Topics
- “Example 2: Managing Entries” on page 166
- “Using PROC DOCUMENT to Work with Table Templates” in SAS Output Delivery System: Advanced Topics

Syntax

```
DIR <path>;
```

Without Arguments
If no options are specified, then the DIR statement displays the current path.

Optional Argument

- **path**
  sets the current directory. 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 directory and the symbol `^^` to represent the parent directory.

DOC Statement
Opens a document and its contents to browse or edit.

Default: Documents are opened in the UPDATE access mode.

Examples:
- “Example 1: Navigating the Directory and Listing the Entries” on page 162
- “Opening and Listing ODS Documents Using WHERE Expressions” in SAS Output Delivery System: Advanced Topics

Syntax

```
DOC <options < access-options>>;
```
Without Arguments

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.

Optional Arguments

**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:

```sql
doc name=yourdoc(write) label='repeated measures results';
run;
```

**Restriction**

A label can be assigned only to documents with Write access permission.

**Requirements**

To use the LABEL= option, specify the NAME= option in the DOC statement.

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-options>**

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-options**

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 is overwritten. If the document does not exist, then it is created.
Interaction  If a label has been specified with the LABEL= option, then it overrides 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 is assigned to the document.

Tip  If the document already exists, then its contents is not changed and the new contents is appended to the document. If the document does not exist, then it is created.

---

**DOC CLOSE Statement**
Closes the current document.

**Syntax**

```
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.

**Syntax**

```
HIDE path <, path-2, ...>;
```

**Required Argument**

`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 directory and the symbol `^^` to represent the parent directory.

---

**IMPORT TO Statement**
Imports the specified SAS data set or graph segment to the specified path.
Example: “Example 4: Importing External Files into an ODS Document” on page 179

Syntax

```
IMPORT DATA= data-set-name<data-set-options> | GRSEG=grseg | TEXTFILE=filename | fileref
TO path </ options>;
```

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 pathname (for example, GRSEG=Sasuser.grseg.mygraph).

See GRSEG= option in 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 directory and the symbol ‘^^’ to represent the parent directory.

**TEXTFILE= <filename | fileref>**
imports a text file into an ODS document that can be replayed to open ODS destinations.

`filename`
specifies the filename. `filename` can be a listing file, a SAS program, or any other text file.

Requirement `filename` 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.

See For information about the FILENAME statement, see SAS DATA Step Statements: Reference

Example “Example 4: Importing External Files into an ODS Document” on page 179

Optional Arguments

**AFTER= path**
imports the data set or graph segment into the directory after the specified path.

Tip You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.
BEFORE= path
imports the data set or graph segment into the directory 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 directory and the symbol "^^" to represent the parent directory.

**data-set-options**
specify actions that apply only to the SAS data set.

See SAS Data Set Options: Reference for information about SAS data sets and their options

**FIRST**
imports the data set or graph segment at the beginning of the directory.

**LAST**
imports the data set or graph segment at the end the directory.

---

**LINK Statement**

Creates a symbolic link from one specified entry to another specified entry.

See: For information about LINK statement advanced options, see "Restructuring Output with PROC DOCUMENT" in SAS Output Delivery System: Advanced Topics.

**Syntax**

```
LINK path TO path </ options>;
```

**Required Argument**

`path`

specifies the locations of the entries that you want to link to one another.

**Tip** You can use the symbol "^" to represent the current directory and the symbol "^^" to represent the parent directory.

**Optional Arguments**

**AFTER= path**

links to the entry that follows the specified path in the current directory.

**BEFORE= path**

links to the entry that precedes the specified path in the current directory.
Tip You can use the symbol `^` to represent the current directory and the symbol `^^` to represent the parent directory.

**FIRST**
links to the first entry in the current directory.

**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:

```sas
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 directory.

---

**LIST Statement**

Lists the contents of one or more entries.

**Defaults:**
Only summary information is displayed if the DETAILS option is omitted.

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 directory, use the LIST statement.

**Examples:**
“Using PROC DOCUMENT to Work with Table Templates” in SAS Output Delivery System: Advanced Topics
“Example 1: Navigating the Directory and Listing the Entries” on page 162
“Opening and Listing ODS Documents Using WHERE Expressions” in SAS Output Delivery System: Advanced Topics
“Example 2: Managing Entries” on page 166
“Example 3: Listing BY-Group Entries” on page 173

**Syntax**

```
LIST path <(where-expression)> <, path-2 <(where-expression-2)>, …> </ options>;
```
Required Argument

**path**
specifies the location of an entry. An entry can be one or more directories, links, or output objects. For example, the following LIST statement lists all of the entries within the Report entry:

```plaintext
list \sasuser.imports\report;
run;
```

**Requirement** Separate multiple paths with commas.

**Tip** You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

Optional Arguments

**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.

**Example** “Example 3: Listing BY-Group Entries” on page 173

**DETAILS**
specifies the properties of the entries. For example, the following LIST statement lists the details of three levels of the Report entry:

```plaintext
list \sasuser.imports\report / details levels=3;
run;
```

**Specifying** list/levels=all details; generates a table with the following columns:

- **Created** shows the date on which the entry was created.
- **Label** shows the entry’s label.
- **Modified** shows the date on which the entry was last modified.
- **Page Break** shows before or after page breaks, if present.
- **Path** shows the entry’s path.
- **Size in Bytes** shows the entry’s size in bites.
- **Symbolic Link** shows a symbolic link associated with the entry, if present.
- **Template** shows the template associated with the entry, if present.
- **Type** shows the entry’s type.
FOLLOW
resolves all links and lists the contents of the entries.

LEVELS= ALL | value
specifies the number of levels that you want to list.

ALL
specifies all levels.

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 directories in ascending order based on the date and time the files were created.

INSERT
lists the directories in the order in which the entries were inserted.

(WHERE=(where-expression-1<operator>))
conditionally selects a subset of entries in an ODS document.

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 value.

SAS function
For information about SAS functions, see SAS Functions and CALL Routines: Reference.

subsetting variable
is a special type 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;
```
_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, suppose that this is the ODS TRACE label path:

Note that in document label paths, the instances of '. ' are replaced with ' /'.

See “ODS TRACE Statement” on page 76

Example The following LIST statement lists all items containing “Fit Statistics” in the label path.

list glm (where={_labelfpath_ ? "Fit Statistics"})/ levels=all;
run;

_MAX_ is the last observation.

Restrictions _MAX_ is used only for output objects.

_MAX_ is used only in the REPLAY statement.

Example The following REPLAY statement replays all observations except the last observation:

replay class(where={_obs_ < _max_});
**_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:

```plaintext
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:

```plaintext
replay ^(where=(mdatetime_='01may04:9:30:00'dt));
run;
```

**_MIN_**

is the first observation.

**Restrictions**

- _MIN_ is used only for output objects.
- _MIN_ is always set to 1
- _MIN_ is used only in the REPLAY statement.

**Example**
The following REPLAY statement replays all observations except the first observation:

```plaintext
replay class(where=(_obs_ < _min_));
```

**_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:

```plaintext
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:

```plaintext
delete glm(where=(name_ ? 'stemleng'));
```

**_OBS_**

is the current observation number in an output object.

**Restrictions**

- _OBS_ is used only for output objects.
- _OBS_ is used only in the REPLAY statement.

**Examples**
The following REPLAY statement replays all but the first ten observations:

```plaintext
replay class(where=(_obs_ > 10));
```
The following REPLAY statement replays all observations except the last observation:
replay class(where=_obs_ < _max_));

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observations:
replay class(where=_obs_ in {1,3,5,7,9});

**observation-number**
is the observation number to be replayed.

**Restrictions**
- **observation-number** is used only for output objects.
- **observation-number** is used only in the REPLAY statement.

**Example**
The following REPLAY statement replays the first, third, fifth, seventh, and ninth observation:
replay class(where=_obs_ in {1,3,5,7,9});

**observation-variable**
is the name of an observation.

**Restrictions**
- **observation-variable** is used only for output objects.
- **observation-variable** is used only in the REPLAY statement.

**Examples**
The following REPLAY statement replays all observations where the variable Weight is greater than 100:
replay class(where=(weight>100));

The following REPLAY statement replays all observations where the variable Sex is equal to 'F'.
replay class(where=(sex='F'));

**_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.

**See**
"Understanding Sequence Numbers " on page 96

**Example**
The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:
replay glm(where=(_seqno_ = 2));

**_TYPE_**
is the type of the current entry.
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:

```plaintext
move ^(where=(_type_ = 'Graph' and _cdate_ = '16JUL2004'd)) to \work.mydoc\monthly;
run;
```

**variable-name** is the name of a BY variable.

The following MOVE TO statement moves all entries where the value of the variable Gender is 'F' to the Monthly directory of Work.Mydoc:

```plaintext
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 6.3  Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ~ or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**Restriction** For the REPLAY statement, the WHERE= option applies to directories and output objects. For the following statements, the WHERE= option applies to directories only:

- COPY TO
- DELETE
- LIST
- MOVE TO

**Requirement** Enclose where-expression in quotation marks.

**See** For advanced information about using where-expressions, see "Using WHERE Expressions in PROC DOCUMENT" in SAS Output Delivery System: Advanced Topics.
You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

Example
“Opening and Listing ODS Documents Using WHERE Expressions” in SAS Output Delivery System: Advanced Topics

MAKE Statement

Creates one or more new directories.

Default: If no location is specified, the newly created directory is appended to the end of the current directory.

Example: “Using PROC DOCUMENT to Work with Table Templates ” in SAS Output Delivery System: Advanced Topics

Syntax

MAKE path <, path-2, ...> </ options>;

Required Argument

path specifies the newly created directory.

Requirement: Separate multiple paths with commas.

Tip: You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

Optional Arguments

AFTER= path
adds the newly created directory after the specified path in the current directory.

Tip: You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

BEFORE= path
adds the newly created directory before the specified path in the current directory.

Tip: You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

FIRST
adds the newly created directory to the beginning of the current directory.

LAST
adds the newly created directory to the end of the current directory.
MOVE TO Statement

Moves entries from the specified location to another location.

Restriction: The root directory cannot be moved or deleted.
Requirement: Separate multiple paths with commas.
Tip: The MOVE TO statement affects all levels of a directory below the specified starting level.

Syntax

```
MOVE path <(where-expression)> <, path-2 <(where-expression-2)>, ...> TO path <(where-options)>;
```

Required Argument

**path** specifies the location of links, output objects, or files that you want to move.

**CAUTION**
The MOVE TO statement affects all levels of a directory below the specified starting level.

Tip You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

Optional Arguments

**AFTER=** path

moves the entry after the specified entry in the path.

Tip You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

**BEFORE=** path

moves the entry before the specified entry in the path.

Tip You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

**FIRST**

moves the entry to the beginning of the specified directory.

**LAST**

moves the entry to the end of the specified directory.

**LEVELS=** ALL | value

specifies the number of levels that you want to move.

ALL

specifies all levels.
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.

(\texttt{WHERE=(where-expression-1<operator >)})
conditionally selects a subset of entries in an ODS document.

\textit{where-expression}
is an arithmetic or logical expression that consists of a sequence of operators
and operands.

\textit{operand}
is one of the following:

\textit{constant}
is a fixed value such as a date literal, a value, or a BY variable value.

\textit{SAS function}
For information about SAS functions, see \textit{SAS Functions and CALL
Routines: Reference}.

\textit{subsetting variable}
is a special type of WHERE expression operand used by the
DOCUMENT procedure to help you find common values in ODS
documents. Here are the subsetting variables:

\texttt{\_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;
```

\texttt{\_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;
```

\texttt{\_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:

```plaintext
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, suppose that this is the ODS TRACE label path:

Note that in document label paths, the instances of '.' are replaced with '\'.

See “ODS TRACE Statement” on page 76

Example The following LIST statement lists all items containing "Fit Statistics" in the label path.

```plaintext
list glm (where=(_labelpath_ ? "Fit Statistics")/ levels=all);
run;
```

_MAX_

is the last observation.

Restrictions _MAX_ is used only for output objects.

_MAX_ is used only in the REPLAY statement.

Example The following REPLAY statement replays all observations except the last observation:

```plaintext
replay class(where=(_obs_ < _max_));
```

_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:

```plaintext
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:

```plaintext
replay ^(where=(_mdatetime_ = '01may04:9:30:00'dt));
run;
```
**_MIN_**

is the first observation.

**Restrictions**

- _MIN_ is used only for output objects.
- _MIN_ is always set to 1
- _MIN_ is used only in the REPLAY statement.

**Example**
The following REPLAY statement replays all observations except the first observation:

```plaintext
replay class(where=(_obs_ < _min_));
```

**_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:

```plaintext
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:

```plaintext
delete glm(where=(_name_ ? 'stemleng'));
```

**_OBS_**

is the current observation number in an output object.

**Restrictions**

- _OBS_ is used only for output objects.
- _OBS_ is used only in the REPLAY statement.

**Examples**
The following REPLAY statement replays all but the first ten observations:

```plaintext
replay class(where=(_obs_ > 10));
```

The following REPLAY statement replays all observations except the last observation:

```plaintext
replay class(where=(_obs_ < _max_));
```

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observations:

```plaintext
replay class(where=(_obs_ in {1,3,5,7,9}));
```

**observation-number**

is the observation number to be replayed.

**Restrictions**

- observation-number is used only for output objects.
- observation-number is used only in the REPLAY statement.

**Example**
The following REPLAY statement replays the first, third, fifth, seventh, and ninth observation:
replay class(where=(_obs_ in {1,3,5,7,9}));

observation-variable
is the name of an observation.

Restrictions  observation-variable is used only for output objects.

observation-variable is used only in the REPLAY statement.

Examples
The following REPLAY statement replays all observations where the variable Weight is greater than 100:
replay class(where=(weight>100));

The following REPLAY statement replays all observations where the variable Sex is equal to 'F'.
replay class(where=(sex='F'));

_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.

See  "Understanding Sequence Numbers " on page 96

Example
The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:
replay glm(where=(_seqno_ = 2));

_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 6.4 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ^= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**Restriction**

For the REPLAY statement, the WHERE= option applies to directories and output objects. For the following statements, the WHERE= option applies to directories only:

- COPY TO
- DELETE
- LIST
- MOVE TO

**Requirement**

Enclose *where-expression* in quotation marks.

**See**

For advanced information about using *where-expressions*, see "Using WHERE Expressions in PROC DOCUMENT" in SAS Output Delivery System: Advanced Topics.

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**

"Opening and Listing ODS Documents Using WHERE Expressions" in SAS Output Delivery System: Advanced Topics

### NOTE Statement

Creates text strings in the current directory.

**Default:**

If you omit the JUST= option, then the note is centered between the left and right margins.

**Example:**

"Example 2: Managing Entries" on page 166
Syntax

**NOTE** `path <'text'> </options>`;

Without Arguments
If no text string is specified, then the NOTE statement creates a blank note.

**Required Argument**

```plaintext
path
```
specifies the location where the note is stored.

**Tip** You can use the symbol `'^'` to represent the current directory and the symbol `'^^'` to represent the parent directory.

**Optional Arguments**

**AFTER=** `path`
inserts the text string after the specified path.

**Tip** You can use the symbol `'^'` to represent the current directory and the symbol `'^^'` to represent the parent directory.

**BEFORE=** `path`
inserts the text string before the specified path.

**Tip** You can use the symbol `'^'` to represent the current directory and the symbol `'^^'` to represent the parent directory.

**FIRST**
inserts the text string at the beginning of the directory.

**JUST=** `LEFT` | `CENTER` | `RIGHT`
specifies the alignment of the text string.

```plaintext
LEFT
```
alines the text string with the left margin.

```plaintext
CENTER
```
centers the text string between the left and right margins.

```plaintext
RIGHT
```
alines the text string with the right margin.

**LAST**
inserts the text string at the end of the directory.

**'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.

Example: “Example 2: Managing Entries” on page 166

Syntax

\texttt{OBANOTE \langle n \rangle \ output-object \langle 'text ' \rangle \langle SHOW\rangle\langle /JUST= \texttt{LEFT | CENTER | RIGHT} \rangle;}

Required Argument

\texttt{output-object}

specifies the name of the ODS output object.

Optional Arguments

\texttt{JUST= \texttt{LEFT | CENTER | RIGHT}}

specifies the alignment of the object footer.

\texttt{LEFT}

aligns the object footer with the left margin.

\texttt{CENTER}

centers the object footer between the left and right margins.

\texttt{RIGHT}

aligns the object footer with the right margin.

\texttt{n}

specifies the relative line that contains the object footer.

Default

If you omit \texttt{n}, then SAS assumes a value of 1. Therefore, specify either \texttt{OBANOTE} or \texttt{OBANOTE1} for the first text line.

Range

1–10

Tips

The OBANOTE line with the highest number appears on the bottom line.

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.

\texttt{SHOW}

specifies that a table containing the output object's heading is written to active destinations.

\texttt{'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 152.

Length
The maximum text length is 32000 characters.

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.

Example: “Example 2: Managing Entries” on page 166

Syntax

OBBNOTE <n> output-object <"text"> <SHOW></JUST= LEFT | CENTER | RIGHT>>;

Required Argument

output-object
specifies the name of the ODS output object.

Optional Arguments

JUST= LEFT | CENTER | RIGHT
specifies the alignment of the object heading.

JUST=

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

Tips
The OBBNOTE line with the highest number appears on the bottom line.

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.
SHOW
specifies that a table containing the output object’s footers is written to active destinations.

'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 152.

**Length**
The maximum text length is 32000 characters.

**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.

---

**OBDYNAM Statement**

Prints a table of dynamic values for the specified output object.

**Example:**

"Example 5: Create a Table of Dynamic Values" on page 183

**Syntax**

OBDYNAM output-object

**Required Argument**

*output-object*
specifies the ODS output object.

**Details**

The output object created by the OBDYNAM statement is a factoid data set named Dynamics, and the description is “Output and data object dynamics”. You can modify the data set and use it with the DYNAMDATA option in the REPLAY statement.

**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.
Example:  
“Example 2: Managing Entries” on page 166

Syntax

```
OBFOOTN <n> output-object <SHOW><'text'>;
```

Required Argument

output-object

specifies the ODS output object.

Optional Arguments

- **n**
  - specifies the relative line that contains the footnote.
  - Range: 1–10

Tips

- 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.
- 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.

SHOW

specifies that a table containing the output object's footnotes is written to active destinations.

' 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 152.

Length

- The maximum text length is 32000 characters.

Requirement

- All text strings must be enclosed in 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.

Example:  
“Example 2: Managing Entries” on page 166
Syntax

OBPAGE output-object < / <DELETE> <AFTER>>;

Without Arguments
If no options are specified, then the OBPAGE statement inserts a page break before an output object.

Required Argument
output-object
specifies the name of the output object.

Optional Arguments
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.

TIP There is an initial page break for each destination for each use of the REPLAY statement. There might be subsequent page breaks depending on the destination and the output being generated.

Restriction Deleting page breaks is not valid in the LISTING destination.

OBSTITLE Statement
Create or modify subtitles.
Example: “Example 2: Managing Entries” on page 166

Syntax

OBSTITLE<n> output-object <text> <SHOW/> <JUST= LEFT | CENTER | RIGHT>;</n

Required Argument
output-object
specifies the ODS output object.
Optional Arguments

**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

**Tips**  
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.

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.

**SHOW**  
specifies that a table containing the output object’s subtitles is written to active destinations.

\( 'text' \)

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 152.

**Length**  
The maximum text length is 32000 characters.

**Requirement**  
All text strings must be enclosed in quotation marks.

**Tip**  
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, if any, 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.

**See:**  
For advanced OBTEMPL statement topics, see “Working with Output Objects and Table Templates” in *SAS Output Delivery System: Advanced Topics*. 
Syntax

\texttt{OBTEMPL output-object;}

Required Argument

\texttt{output-object}

specifies the pathname of the output object.

See \textit{“Getting Familiar with Output Objects” on page 96}

Example \textit{“Example 3: Listing BY-Group Entries” on page 173}

\section*{OBTITLE Statement}

Creates or modifies title lines for the output.

\textbf{Tip:} The OBTITLE is similar to the global TITLE statement.

\textbf{Example:} \textit{“Example 2: Managing Entries” on page 166}

Syntax

\texttt{OBTITLE<n> output-object \langle SHOW\rangle \langle text \rangle;}

Required Argument

\texttt{output-object}

specifies the name of the output object.

Optional Arguments

\texttt{n}

specifies the relative line that contains the title.

Range \hspace{1em} 1–10

\textbf{Tips}

The OBTITLE line with the highest number appears on the bottom line. If you omit \texttt{n}, then SAS assumes a value of 1. Therefore, specify OBTITLE or OBTITLE1 for the first text line.

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.

\texttt{SHOW}

specifies that a table containing the output object’s titles is written to active destinations.
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 152.

Length

The maximum text length is 32000 characters.

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

Renames an entry.

Syntax

RENAME path-1 TO path-2;

Required Arguments

path-1

specifies the current directory or output object.

Tip

You can use the symbol "^" to represent the current directory and the symbol "^^" to represent the parent directory.

path-2

specifies the new name of the directory or output object.

Tip

You can use the symbol "^" to represent the current directory and the symbol "^^" to represent the parent directory.

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.

Restrictions:

User-defined format names must be unique within an ODS DOCUMENT.

When replaying an ODS document, values created by the MVAR statement must be re-created in the same session that is replaying the document.
Tip: There is an initial page break for each destination for each use of the REPLAY statement. There might be subsequent page breaks depending on the destination and the output being generated.

Examples: “Example 2: Managing Entries” on page 166
“Using PROC DOCUMENT to Work with Table Templates ” in SAS Output Delivery System: Advanced Topics

Syntax

REPLAY path <(where-expression)> <, path-2 <(where-expression-2)>, …> </options>;

Optional Arguments

ACTIVEFOOTN
specifies that footnotes that are active in a SAS session override the footnotes that are stored in an ODS document.

Alias ACFOOTN

ACTIVETITLE
specifies that titles that are active in a SAS session override the titles that are stored in an ODS document.

Alias ACTITLE

ANCESTORS=n | ALL
specifies the number of parent directories to replay.

n
specifies the number of parent directories to replay. n must be a nonnegative integer.

ALL
specifies that all directories are replayed.

Default ALL

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, enclose the ODS destinations in 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 For information about ODS destinations, see “Understanding ODS Destinations” in SAS Output Delivery System: User’s Guide.
DYNAMDATA=\textit{data-set-name}

specifies the name of the data set that contains the dynamic values.

\textbf{See} For information about how to create a table of dynamics, see "\textit{OBDYNAM Statement}" on page 135.

\textbf{LEVELS=} \textbf{ALL} | \textit{value}

specifies the number of levels that you want to replay.

\textit{ALL}

specifies that all levels of the directory are displayed to all open destinations.

\textit{value}

specifies the numeric value of the path level. For example, the following \texttt{REPLAY} statement replays two levels of the entry Data to the HTML and RTF destinations:

\begin{verbatim}
replay \Report\GLM#1\Data / levels=2 dest=(html rtf); run;
\end{verbatim}

\textbf{Default} \textbf{ALL}

\textbf{Restriction} The \textbf{LEVELS=} option is valid only when you specify a directory.

\textbf{path}

specifies the location of an entry. An entry can be one or more directories, links, or output objects.

\textbf{Requirement} Separate multiple paths with commas.

\textbf{Tip} You can use the symbol '^' to represent the current directory and the symbol '^^' to represent the parent directory.

\textbf{STORE=}\textit{template-store}

specifies the template store when replaying output objects from the path. If the replayed path is an output object, the template store applies only to that output object. If the path is a directory, the template store applies to any output objects contained within the directory.

\textbf{Restriction} The \textbf{STORE=} option applies only to output objects with external templates. Output objects with internal templates, such as those generated by \texttt{PROC PRINT}, \texttt{PROC REPORT}, \texttt{PROC TABULATE}, and \texttt{PROC SGPLOT}, are unaffected by \textbf{STORE=}.

\textbf{Requirement} The \textbf{STORE=} option can be used only if the \textit{path} option is also specified in the \texttt{REPLAY} statement.

\textbf{Tip} The \textbf{STORE=} option is useful if you want to use different templates with a single or multiple output objects.

\textbf{Example} "Using \texttt{PROC DOCUMENT} to Work with Table Templates" in \textit{SAS Output Delivery System: Advanced Topics}

\texttt{(\texttt{WHERE=\langle where-expression-1<operator>\rangle})}

conditionally selects a subset of entries in an ODS document.

\textit{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 value.

SAS function

For information about SAS functions, see SAS Functions and CALL Routines: Reference.

subsetting variable

is a special type 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, suppose that this is the ODS TRACE label path:
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 glm (where=(\_labelpath_ ? "Fit Statistics"))/ levels=all;
run;
```

_\_MAX_ is the last observation.

**Restrictions**

_\_MAX_ is used only for output objects.

_\_MAX_ is used only in the REPLAY statement.

**Example**
The following REPLAY statement replays all observations except the last observation:

```
replay class(where=(_obs_ < _max_));
```

_\_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;
```

_\_MIN_ is the first observation.

**Restrictions**

_\_MIN_ is used only for output objects.

_\_MIN_ is always set to 1

_\_MIN_ is used only in the REPLAY statement.

**Example**
The following REPLAY statement replays all observations except the first observation:

```
replay class(where=(_obs_ < _min_));
```

_\_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'));
```

_OBS_

is the current observation number in an output object.

Restrictions

_OBS_ is used only for output objects.

_OBS_ is used only in the REPLAY statement.

Examples

The following REPLAY statement replays all but the first ten observations:

```
replay class(where=(obs_ > 10));
```

The following REPLAY statement replays all observations except the last observation:

```
replay class(where=(obs_ < _max_));
```

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observations:

```
replay class(where=(obs_ in (1,3,5,7,9)));
```

observation-number

is the observation number to be replayed.

Restrictions

observation-number is used only for output objects.

observation-number is used only in the REPLAY statement.

Example

The following REPLAY statement replays the first, third, fifth, seventh, and ninth observation:

```
replay class(where=(obs_ in (1,3,5,7,9)));
```

observation-variable

is the name of an observation.

Restrictions

observation-variable is used only for output objects.

observation-variable is used only in the REPLAY statement.

Examples

The following REPLAY statement replays all observations where the variable Weight is greater than 100:

```
replay class(where=(weight>100));
```

The following REPLAY statement replays all observations where the variable Sex is equal to 'F'.

```
replay class(where=(sex='F'));
```

_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:

```plaintext
list ^(where=(_path_ ? 'Anova'));
run;
```

_**SEQNO**_  is the sequence number of the current entry.

See  "Understanding Sequence Numbers " on page 96

Example  The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:

```plaintext
replay glm(where=(_seqno_ = 2));
```

_**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:

```plaintext
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:

```plaintext
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 6.5  Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or =~ or ~= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>Symbol</td>
<td>Mnemonic Equivalent</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------</td>
<td>------------</td>
</tr>
<tr>
<td>IN</td>
<td>Equal to one from a list of values</td>
<td></td>
</tr>
</tbody>
</table>

**Restriction**
For the REPLAY statement, the WHERE= option applies to directories and output objects. For the following statements, the WHERE= option applies to directories only:
- COPY TO
- DELETE
- LIST
- MOVE TO

**Requirement**
Enclose *where-expression* in quotation marks.

**See**
For advanced information about using *where-expressions*, see “Using WHERE Expressions in PROC DOCUMENT” in *SAS Output Delivery System: Advanced Topics*.

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Data Set Options: Reference* and the section on WHERE-Expression Processing in *SAS Language Reference: Concepts*.

**Example**
“Opening and Listing ODS Documents Using WHERE Expressions” in *SAS Output Delivery System: Advanced Topics*

---

**SETLABEL Statement**
Assigns a label to the specified path.

**Example:**
“Using PROC DOCUMENT to Work with Table Templates ” in *SAS Output Delivery System: Advanced Topics*

**Syntax**

```plaintext
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.
For more information, see “Customizing Labels, Titles, and Footnotes with BY Variables” on page 152.

**path**
specifies the location of a link, output object, or directory.

**Tip** You can use the symbol ‘^’ to represent the current directory and the symbol ‘^^’ to represent the parent directory.

---

### UNHIDE Statement

Enables the output of a hidden entry to be displayed when it is replayed.

**Syntax**

```
UNHIDE path <, path-2, ... path-n>
```

**Required Argument**

**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 directory and the symbol ‘^^’ to represent the parent directory.

---

### Usage: DOCUMENT Procedure

#### Working with ODS Documents

**Creating an ODS Document**

An ODS document is a hierarchical file of output objects that is created from a procedure or data query. The ODS document holds these output objects in their original structures, but you can rearrange the hierarchy and structure of these objects. ODS documents are stored in a proprietary format (a document store) and can be viewed only with SAS software. To view or modify what is in the document store, you must use either the Documents window in the SAS Windowing Environment or the DOCUMENT procedure.
To create an ODS document, you must use the Documents window in the SAS Windowing Environment or the “ODS DOCUMENT Statement”. The following code creates the ODS document Work.Prddoc within a document store:

```sas
proc sort data=sashelp.prdsale out=prdsale;
  by Country;
run;
ods document name=work.prddoc(write);
proc tabulate data=prdsale;
  by Country;
  var predict;
  class prodtype;
  table prodtype all,
       predict*(min mean max);
run;
ods select ExtremeObs;
proc univariate data=prdsale;
  by Country;
  var actual;
run;
ods document close;
```

The following display shows the properties of the first Table 1, which is the summary report for Canada. To view the properties of an entry, from the SAS Windowing Environment, right-click on the highlighted object and select Properties from the pop-up list. In the Table Properties window, you can see the document name and the document path, among other information.

Figure 6.1 Table Properties for Table 1
This ODS document was written to the Work library. However, if it had been written to a permanent location (such as c:\temp\output), you would see in Windows Explorer that the document store has a file extension of SAS7BITM.

Viewing the Contents of an ODS Document

After you have created a document with the ODS DOCUMENT statement, you can use PROC DOCUMENT to view the contents of your document. You can rearrange, duplicate, or remove output from the results of a procedure or a database query without invoking the procedures from the original report. The first step is to view your document’s contents by using the LIST statement. The LIST statement enables you to view the object list and folder structure within the ODS document. The following code creates a list of all levels of the document Work.Prddoc:

```sas
proc document name=work.prddoc;
   list / levels=all;
run;
quit;
```

The LIST statement can be used to list what is in an entire document or just one of the entries. For more information about the LIST statement, see “LIST Statement” on page 118.

**TIP** In the SAS Windowing Environment, every folder icon in the Results window corresponds to an item in the LIST statement output.
### Output 6.1 Data Structure of the Work.Prddoc ODS Document

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tabulate#1</td>
<td>Dir</td>
</tr>
<tr>
<td>2</td>
<td>Tabulate#1\ByGroup#1</td>
<td>Dir</td>
</tr>
<tr>
<td>3</td>
<td>Tabulate#1\ByGroup#1\Report#1</td>
<td>Dir</td>
</tr>
<tr>
<td>4</td>
<td>Tabulate#1\ByGroup#1\Report#1\Table#1</td>
<td>Table</td>
</tr>
<tr>
<td>5</td>
<td>Tabulate#1\ByGroup#2#1</td>
<td>Dir</td>
</tr>
<tr>
<td>6</td>
<td>Tabulate#1\ByGroup#2#1\Report#1</td>
<td>Dir</td>
</tr>
<tr>
<td>7</td>
<td>Tabulate#1\ByGroup#2#1\Report#1\Table#1</td>
<td>Table</td>
</tr>
<tr>
<td>8</td>
<td>Tabulate#1\ByGroup#3#1</td>
<td>Dir</td>
</tr>
<tr>
<td>9</td>
<td>Tabulate#1\ByGroup#3#1\Report#1</td>
<td>Dir</td>
</tr>
<tr>
<td>10</td>
<td>Tabulate#1\ByGroup#3#1\Report#1\Table#1</td>
<td>Table</td>
</tr>
<tr>
<td>11</td>
<td>Univariate#1</td>
<td>Dir</td>
</tr>
<tr>
<td>12</td>
<td>Univariate#1\ByGroup#1#1</td>
<td>Dir</td>
</tr>
<tr>
<td>13</td>
<td>Univariate#1\ByGroup#1\ACTUAL#1</td>
<td>Dir</td>
</tr>
<tr>
<td>14</td>
<td>Univariate#1\ByGroup#1\ACTUAL#1\ExtremeObs#1</td>
<td>Table</td>
</tr>
<tr>
<td>15</td>
<td>Univariate#1\ByGroup#2#1</td>
<td>Dir</td>
</tr>
<tr>
<td>16</td>
<td>Univariate#1\ByGroup#2#1\ACTUAL#1</td>
<td>Dir</td>
</tr>
<tr>
<td>17</td>
<td>Univariate#1\ByGroup#2#1\ACTUAL#1\ExtremeObs#1</td>
<td>Table</td>
</tr>
<tr>
<td>18</td>
<td>Univariate#1\ByGroup#3#1</td>
<td>Dir</td>
</tr>
<tr>
<td>19</td>
<td>Univariate#1\ByGroup#3#1\ACTUAL#1</td>
<td>Dir</td>
</tr>
<tr>
<td>20</td>
<td>Univariate#1\ByGroup#3#1\ACTUAL#1\ExtremeObs#1</td>
<td>Table</td>
</tr>
</tbody>
</table>

### Items Included in an ODS Document

In an ODS document, each level of the hierarchical file represents a path that refers to the location of a file, link, or output object. An output object can be one of the following:

- table
- graph
- equation
- note
- SAS/GRAPH external graph titles
Items Not Included in an ODS Document

An ODS document does not store the following items:

- GRSEGs (References to GRSEGs, but not GRSEGs themselves, are stored.)
- ODS options
- procedure options
- Report Writing Interface output
- SAS logs
- SAS/GRAPH options
- SAS system options

ODS Document Persistence

An ODS document persists in the SAS system until the document, or the SAS library containing the document, is deleted. An ODS document that was created in the Sasuser library, or in another permanent SAS library, can persist indefinitely. 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.

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


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 133
- “OBBNOTE Statement” on page 134
Note: The #BYVAL, #BYVAR, and #BYLINE substitutions show up only for 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\textit{n} | #BYVAL\textit{(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 non-alphanumeric 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:

  \begin{itemize}
  \item \texttt{n}
  \end{itemize}

  specifies which variable in the BY statement #BYVAL should use. The value of \texttt{n} indicates the position of the variable in the BY statement.

Example

#BYVAL2 specifies the second variable in the BY statement.

\texttt{variable-name}

names the BY variable.

\textbf{Requirement} You must enclose \texttt{variable-name} in parentheses.

\textbf{Tip} \texttt{variable-name} is not case sensitive.

\textbf{Example}

#BYVAL(YEAR) specifies the BY variable, YEAR.

#BYVAR\textit{n} | #BYVAR\textit{(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 non-alphanumeric 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.

Requirement  You must enclose variable-name in parentheses.

Tip  variable-name is not case sensitive.

Example  #BYVAR(SITES) specifies the BY variable SITES.

#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 the Results Window in the SAS Windowing Environment

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 when 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 DOCUMENT Procedure” on page 158.
Viewing Entries in the Results Window

To view the Results window, submit this command in the command bar:

```plaintext
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.

![Figure 6.2 Results Window Showing the Output Object “Normal” in Three Formats](image)

For more information about using the Results window, make the Results window the active window and select Help ➔ Using This Window.

Using the Documents Window in the SAS Windowing Environment

Viewing an ODS Document in the Documents Window

To view the Documents window, submit the following command in the command bar:

```plaintext
odsdocuments
```

This display shows the Documents window that contains the ODS document named **Sasuser.Univ**. In the display, notice that **Sasuser.Univ** contains several directory levels. The **Exponential_x** directory 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.
The Documents window contains these items:

**entry**
- is an output object, link, or directory.

**Note:** Only output objects of the type Document are displayed in the Documents window.

**directory**
- 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. The Results window and the Documents window use this icon to indicate an ODS document output object.

**Understanding When to Use the Documents Window**

In the SAS Windowing Environment, 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
<table>
<thead>
<tr>
<th>Task</th>
<th>Menu Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a new ODS destination output type</td>
<td>Open As</td>
</tr>
<tr>
<td>Show the entries that were previously</td>
<td>Show Excluded</td>
</tr>
<tr>
<td>excluded</td>
<td></td>
</tr>
<tr>
<td>Remove from the tree, but do not delete</td>
<td>Exclude</td>
</tr>
<tr>
<td>the selected entry</td>
<td></td>
</tr>
<tr>
<td>Replay the selected entry to all open ODS</td>
<td>Replay</td>
</tr>
<tr>
<td>destinations</td>
<td></td>
</tr>
</tbody>
</table>

Replaying a Document Using the Document Window

You can use the Documents window to replay a document. Replaying your document keeps you from having to rerun your procedure steps.

To replay an entire document, right-click on the document and select **Open As** from the drop-down menu. When the Open As window appears, select a destination, such as PDF, and then click **OK**. You can select multiple destinations by pressing the Ctrl key and selecting the destinations that you want.

When you use the Open As window, SAS creates a default name. Using PROC DOCUMENT and the REPLAY statement enables you to specify a name for your ODS output when you replay the document.
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 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 6.7 Tasks That You Can and Cannot Do in the Documents Window and with the DOCUMENT Procedure

<table>
<thead>
<tr>
<th>Task</th>
<th>Documents Window</th>
<th>DOCUMENT Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new ODS document</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create a new folder</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Import a data set or graph segment</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Copy folders or output objects</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Move folders or output objects</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create a symbolic link from one output object to another output object</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Delete a document, folder, or output object</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rename a folder or output object</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Assign a description to a folder or output object</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Task</td>
<td>Documents Window</td>
<td>DOCUMENT Procedure</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Prevent entries from being displayed when they are replayed</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Show entries that are excluded</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enable hidden entries to be displayed</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Replay to the specified open ODS destinations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Determine the path specification</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Set or display the current directory</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or delete a page break</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or modify title lines</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or modify subtitles</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or modify the lines of text before output objects</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or modify the lines of text after output objects</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or modify footnote lines</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create text strings in the current folder</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Comparisons between the Documents Window and the Results Window in the SAS Windowing Environment

**Table 6.8  Tasks That You Can and Cannot Do in the Documents Window and the Results Window**

<table>
<thead>
<tr>
<th>Task</th>
<th>Documents window</th>
<th>Results window</th>
</tr>
</thead>
<tbody>
<tr>
<td>View all SAS documents including those stored in SAS libraries</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>View output object types that are created when you run a SAS program, such as HTML, PDF, and SAS document</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>View the results after you create a new output object</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Customize the layout of output objects</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>View the property information of SAS documents</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>View the properties of an output object</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Delete or rename entries</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Copy or move SAS documents</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Create shortcuts to SAS documents</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Drag and drop output objects</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Viewing the Properties of an Entry in the SAS Windowing Environment

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.


**Figure 6.4 Entry Properties Window**

Items vary, depending on the entry that you select in the Documents or Results windows. The Results window for an ODS document output object can contain these items:

- **Created**
  - is the date on which 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**
  - is the location of the entry in the tree structure. If you move the entry to another location in the Documents window, then this path changes.

- **Modified**
  - is the date on which 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.

**Advanced ODS DOCUMENT Features**

The ODS DOCUMENT destination enables you to store the output objects from your procedure or process in a special type of item store called a document store. With the DOCUMENT procedure, you are not limited to simply regenerating the same report. You can change the order in which objects are rendered, the table of contents, the templates that are used, macro variables, and ODS and system options. The following sections show you some of the more advanced features of PROC DOCUMENT.
Examples: DOCUMENT Procedure

Example 1: Navigating the Directory and Listing the Entries

Features:
- ODS DOCUMENT statement option
  - NAME=
- DOC statement option
  - NAME=
- LIST statement options
  - entry
    - LEVELS=
    - DETAILS
- DIR statement option
  - path
- Procedure output
  - PROC DOCUMENT

ODS destinations:
- DOCUMENT, HTML

Details

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 directories

Program

```sas
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;
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\Exponential_x\Histogram\Exponential;
  list;
  list fitquantiles/details;
run;
quit;

Program Description

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 is 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 about all levels of the document Work.Univ.

   proc document;
     doc;
     doc name=univ;
     list/levels=all;

Set the current directory to EXPONENTIAL, list the contents of the EXPONENTIAL directory, select a table, and list the details of the table that you selected. The DIR statement changes the current directory to \Univariate#2\Exponential_x#1\Histogram#1\Exponential#1. The path \Univariate#2\Exponential_x#1\Histogram#1\Exponential#1 was obtained from the listing of the Work.Univ document. The LIST statement (with no arguments) lists the contents of EXPONENTIAL. The LIST FITQUANTILES/DETAILS statement specifies that ODS opens the FitQuantiles table and lists its details.

   dir \Univariate#2\Exponential_x#1\Histogram#1\Exponential#1;
   list;
   list fitquantiles/details;
run;

**Terminate the DOCUMENT procedure.** Specify the QUIT statement to terminate the DOCUMENT procedure. If you omit QUIT, then you cannot view DOCUMENT procedure output.

quit;

**Output**

**Output 6.2**  List of the Entries of the ODS Document Work.Univ and the Properties of Those Entries

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\Univariate#1</td>
<td>Dir</td>
</tr>
<tr>
<td>2</td>
<td>\Univariate#1\Normal_x#1</td>
<td>Dir</td>
</tr>
<tr>
<td>3</td>
<td>\Univariate#1\Normal_x#1\Histogram#1</td>
<td>Dir</td>
</tr>
<tr>
<td>4</td>
<td>\Univariate#1\Normal_x#1\Histogram#1\Histogram#1</td>
<td>Graph</td>
</tr>
<tr>
<td>5</td>
<td>\Univariate#2</td>
<td>Dir</td>
</tr>
<tr>
<td>6</td>
<td>\Univariate#2\Exponential_x#1</td>
<td>Dir</td>
</tr>
<tr>
<td>7</td>
<td>\Univariate#2\Exponential_x#1\Histogram#1</td>
<td>Dir</td>
</tr>
<tr>
<td>8</td>
<td>\Univariate#2\Exponential_x#1\Histogram#1\Histogram#1</td>
<td>Graph</td>
</tr>
<tr>
<td>9</td>
<td>\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1</td>
<td>Dir</td>
</tr>
<tr>
<td>10</td>
<td>\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\ParameterEstimates#1</td>
<td>Table</td>
</tr>
<tr>
<td>11</td>
<td>\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\GoodnessOfFit#1</td>
<td>Table</td>
</tr>
<tr>
<td>12</td>
<td>\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\FitQuantiles#1</td>
<td>Table</td>
</tr>
</tbody>
</table>

**Output 6.3**  List of the Entries of the Exponential#1 Entry and the Properties of Those Entries

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ParameterEstimates#1</td>
<td>Table</td>
</tr>
<tr>
<td>2</td>
<td>GoodnessOfFit#1</td>
<td>Table</td>
</tr>
<tr>
<td>3</td>
<td>FitQuantiles#1</td>
<td>Table</td>
</tr>
</tbody>
</table>
Example 2: Managing Entries

Features:
- PROC DOCUMENT statement option
  - NAME=
- DIR statement
- LIST statement option
  - LEVELS=
- NOTE statement
- OBANOTE statement option
  - SHOW
- OBBNOTE statement option
  - SHOW
- OBFOOTN statement option
  - SHOW
- OBPAGE statement
- OBSTITLE statement option
  - SHOW
- OBTITLE statement option
  - SHOW
- REPLAY statement
- Procedure output
  - PROC CONTENTS

ODS destinations:
- DOCUMENT, HTML

Details

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
specifies that a table containing the output object’s headings, titles, and footnotes, be written to the active destinations

generate a listing of the entries that shows the details for each entry

Program

ods document name=class(write);
title 'Title Specified by the Global TITLE Statement';
footnote 'Footnote Specified by the Global FOOTNOTE Statement';
ods document close;
proc document name=class;
   dir \Contents\DataSet; 
run;
   obtitle Attributes 'Title Specified by the OBTITLE Statement';
run;
quit;
proc document name=class;
   dir \Contents\DataSet; 
run;
   obbnote Attributes 'Object Heading Note Specified by the OBBNOTE Statement';
run;
quit;
proc document name=class;
   dir \Contents\DataSet; 
run;
   obfootn Variables 'Change the Global Footnote with the OBFOOTN Statement';
run;
quit;
proc document name=class;
   dir \Contents\DataSet; 
run;
   obanote Attributes 'Object Footer Note Specified by the OBANOTE Statement';
run;
quit;
proc document name=class;
   dir \Contents\DataSet; 
run;
   obstitle Attributes 'Subtitle Specified by the OBSTITLE Statement';
run;
quit;
proc document name=class;
   note addnote 'Note added to the document';
run;
quit;
ods html file='your_file.html' style=Sapphire;
proc document name=class;
   obpage \Contents\DataSet\Variables; 
replay;
run;
quit;
proc document name=class;
Program Description

Create the Class document. The NAME= option creates an ODS document named Class. The CONTENTS procedure shows the contents of the SAS data set Sashelp.Class. The entries in the ODS document Class are used in the remainder of this example.

```sas
ods document name=class(write);
title 'Title Specified by the Global TITLE Statement';
footnote 'Footnote Specified by the Global FOOTNOTE Statement';
ods document close;
```

Change the global title. The OBTITLE statement assigns a new title to the Attributes#1 entry. The NAME= option specifies the current ODS 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.

```sas
proc document name=class;
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. The NAME= option specifies the current ODS document. The DIR statement changes the current directory to \Contents#1\DataSet#1. The QUIT statement terminates PROC DOCUMENT.

```sas
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 directory to \Contents#1\DataSet#1. The QUIT statement terminates PROC DOCUMENT.

```sas
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. The NAME= option specifies the current ODS document. The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
  dir \Contents#1\DataSet#1;
  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 NAME= option specifies the current ODS document. The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
  dir \Contents#1\DataSet#1;
  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.

```plaintext
proc document name=class;
  note addnote 'Note added to the document';
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 Sapphire. 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.

```plaintext
ods html file='your_file.html' style=Sapphire;
proc document name=class;
  obpage \Contents#1\DataSet#1\Variables#1;
  replay;
run;
quit;
```

Generate a table containing contextual information for each title, footnote, and note. 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.

```plaintext
proc document name=class;
  list/ levels=all details;
  dir \Contents#1\DataSet#1;
  obanote Attributes#1 show;
```
Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination.

ods html close;
Output

Output 6.5  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 OBNOTE Statement

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>SASHELP.CLASS</th>
<th>Observations</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>5</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>09/23/2018 22:38:57</td>
<td>Observation Length</td>
<td>40</td>
</tr>
<tr>
<td>Last Modified</td>
<td>09/23/2018 22:38:57</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
<td>Compressed</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td></td>
<td>Sorted</td>
<td>NO</td>
</tr>
<tr>
<td>Label</td>
<td>Student Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>WINDOWS_64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>us-ascii ASCII (ANSI)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Object Footer Note Specified by the OBANOTE Statement

<table>
<thead>
<tr>
<th>Data Set Page Size</th>
<th>65536</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Rows</td>
<td></td>
</tr>
</tbody>
</table>

Change the Global Footnote with the OBFOOTN Statement

Title Specified by the Global TITLE Statement

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Age</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Height</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>Name</td>
<td>Char</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Sex</td>
<td>Char</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Weight</td>
<td>Num</td>
<td>8</td>
</tr>
</tbody>
</table>

Note added to the document
### Output 6.6  Listing of Work.Class

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
<th>Size in Bytes</th>
<th>Created</th>
<th>Modified</th>
<th>Symbolic Link</th>
<th>Template</th>
<th>Label</th>
<th>Page Break</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\Contents#1</td>
<td>Dir</td>
<td></td>
<td>27SEP2018:09:01:22</td>
<td>27SEP2018:09:01:22</td>
<td></td>
<td>The Contents Procedure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>\Contents#1\DataSel#1</td>
<td>Dir</td>
<td></td>
<td>27SEP2018:09:01:23</td>
<td>27SEP2018:09:01:23</td>
<td></td>
<td>SASHELP.CLASS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>\Contents#1\DataSel#1\Attributes#1</td>
<td>Table</td>
<td>694</td>
<td>27SEP2018:09:01:22</td>
<td>27SEP2018:09:01:22</td>
<td></td>
<td>Base.Contents.Attributes</td>
<td>Attributes</td>
<td>Below</td>
</tr>
<tr>
<td>4</td>
<td>\Contents#1\DataSel#1\EngineHost#1</td>
<td>Table</td>
<td>707</td>
<td>27SEP2018:09:01:22</td>
<td>27SEP2018:09:01:22</td>
<td></td>
<td>Base.Contents.EngineHost</td>
<td>EngineHost Information</td>
<td>Below</td>
</tr>
<tr>
<td>5</td>
<td>\Contents#1\DataSel#1\Variables#1</td>
<td>Table</td>
<td>341</td>
<td>27SEP2018:09:01:22</td>
<td>27SEP2018:09:01:22</td>
<td></td>
<td>Base.Contents.Variables</td>
<td>Variables</td>
<td>Below</td>
</tr>
<tr>
<td>6</td>
<td>\Contents#1\DataSel#1\addnote#1</td>
<td>Note</td>
<td>214</td>
<td>27SEP2018:09:01:22</td>
<td>27SEP2018:09:01:22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>addnote#1</td>
<td>Note</td>
<td>214</td>
<td>27SEP2018:09:01:42</td>
<td>27SEP2018:09:01:42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Footnote Specified by the Global FOOTNOTE Statement
Example 3: Listing BY-Group Entries

Features: LIST statement options
      BYGROUPS
      LEVELS
      LIST
      PROC DOCUMENT statement option
      NAME=
      OBTEMPL statement
      Procedure output
PROC DOCUMENT
PROC STANDARD

ODS destinations:
DOCUMENT, HTML

Details
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
ods document name=mydocument(write);
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;
proc standard mean=80 std=5 out=StndScore print;
  by section student;
  var stest1-stest3;
run;
ods document close;
proc document name=mydocument;
title "Listing of MyDocument Using the BYGROUPS Option";
run;
  list/ levels=all bygroups;
Program Description

Create the ODS document MyDocument and open the DOCUMENT destination. 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 is created in the Work library. Assign a libref to create MyDocument in a permanent library.

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.

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 is displayed. 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 about 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. The OBTEMPL statement writes the table template that is associated with the output object Standard#1 to the HTML destination. The ODS DOCUMENT CLOSE statement closes the DOCUMENT destination. If the DOCUMENT destination is not closed, no DOCUMENT procedure output can be viewed. If you omit LEVELS=ALL, then no entry list is 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.

To see what the output looks like if you omit the BYGROUPS option, see Output 6.18 on page 178.

    proc document name=mydocument;
    title "Listing of MyDocument Using the BYGROUPS Option";
    run;
    list/ levels=all bygroups;
    run;
    title "Table Template for the Output Object Standard#1";
    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 cannot view DOCUMENT procedure output.

    quit;

Output

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.
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 6.9  Listing of Work.MyDocument with the BYGROUPS Option Specified

- **Listing of MyDocument Using the BYGROUPS Option**
  - **Order by:** Insertion
  - **Number of levels:** All

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
<th>Section</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\Standard#1\ByGroup1#1\Standard#1</td>
<td>Table</td>
<td>1.00000</td>
<td>238900645</td>
</tr>
<tr>
<td>2</td>
<td>\Standard#1\ByGroup2#1\Standard#1</td>
<td>Table</td>
<td>1.00000</td>
<td>242760674</td>
</tr>
<tr>
<td>3</td>
<td>\Standard#1\ByGroup3#1\Standard#1</td>
<td>Table</td>
<td>1.00000</td>
<td>254701167</td>
</tr>
<tr>
<td>4</td>
<td>\Standard#1\ByGroup4#1\Standard#1</td>
<td>Table</td>
<td>1.00000</td>
<td>263924860</td>
</tr>
<tr>
<td>5</td>
<td>\Standard#1\ByGroup5#1\Standard#1</td>
<td>Table</td>
<td>1.00000</td>
<td>999001230</td>
</tr>
<tr>
<td>6</td>
<td>\Standard#1\ByGroup6#1\Standard#1</td>
<td>Table</td>
<td>2.00000</td>
<td>238806445</td>
</tr>
<tr>
<td>7</td>
<td>\Standard#1\ByGroup7#1\Standard#1</td>
<td>Table</td>
<td>2.00000</td>
<td>416724915</td>
</tr>
<tr>
<td>8</td>
<td>\Standard#1\ByGroup8#1\Standard#1</td>
<td>Table</td>
<td>2.00000</td>
<td>459700866</td>
</tr>
<tr>
<td>9</td>
<td>\Standard#1\ByGroup9#1\Standard#1</td>
<td>Table</td>
<td>2.00000</td>
<td>990001252</td>
</tr>
<tr>
<td>10</td>
<td>\Standard#1\ByGroup10#1\Standard#1</td>
<td>Table</td>
<td>2.00000</td>
<td>999002527</td>
</tr>
</tbody>
</table>
Example 4: Importing External Files into an ODS Document

Features:
- PROC DOCUMENT statement option
  - NAME=
- IMPORT statement option
  - TEXTFILE=
- LIST statement
- REPLAY statement
- Procedure output
  - PROC DOCUMENT
  - PROC GLM

ODS destinations:
- DOCUMENT, HTML, LISTING
Details

The following example uses the IMPORT TO statement to import two files into an ODS document. The first file contains LISTING output, and the second file contains a SAS program. The files are imported into an ODS document and then replayed to a PDF document.

In this example, the SAS program that is imported is the entire example itself, which was saved as textfileExample.sas. Before you run this example, save the program in an external file named textfileExample.sas.

Program

```
proc glm 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;
run;
```

```
title1 'Importing a SAS Program and LISTING Output';
data one;
  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;
ods listing file="your-file-path/odsglm.lst";
proc glm;
  class month;
  model age age2 age3=month / nouni; manova h=month /printe;
run;
quit;
```

```
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;
proc glm 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;
ods listing close;
ods listing;
proc document name=import(write);
import textfile="your-file-path\odsglm.lst" to ^;
import textfile="your-file-path\textfileExample.sas" to ^;
list/ details;
run;
ods pdf file="out.pdf";
replay;
run;
quit;
ods pdf close;

Program Description

Add a title. The TITLE statement specifies a title.

title1 'Importing a SAS Program and LISTING Output';

Create the ONE data set.

data one;
  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;

Create the listing file to be imported. The ODS LISTING statement creates the file Odsglm.lst, which contains the LISTING output.

ods listing file="your-file-path/odsglm.lst";

Run the GLM procedure.

proc glm;
  class month;
  model age age2 age3=month / nouni; manova h=month /printe;
run;
quit;

Create the Plants data set.

data plants;
  input type $ @;
  do block=1 to 3;
    input stemleng @;
    output;
  end;
run;
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 the GLM procedure.

proc glm 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;
ods listing;

Run the DOCUMENT procedure. The PROC DOCUMENT statement names the
document "Import" and opens it for Write access. The first IMPORT TO statement
with the TEXTFILE= option specified statement imports the file Odsglm.lst to the
ODS document in the current directory. The second IMPORT TO statement with the
TEXTFILE= option specified statement imports the file textfileExample.sas to the
ODS document in the current directory.

ods listing;
proc document name=import(write);
  import textfile="your-file-path\odsglm.lst" to ^;
  import textfile="your-file-path\textfileExample.sas" to ^;
  list/ details;
run;

Replay the document to a PDF file. The REPLAY statement replays the document
to the PDF file Out.pdf.

ods pdf file="out.pdf";
  replay;
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 cannot view DOCUMENT procedure output. The ODS PDF CLOSE statement
closes the PDF destination.

quit;
ods pdf close;
Example 5: Create a Table of Dynamic Values

Features:
- PROC DOCUMENT statement option
  NAME=
- DIR statement
- DOC statement
- LIST statement
- OBDYNAM statement
- Procedure output
- PROC DOCUMENT
- PROC TABULATE

ODS destinations:
- DOCUMENT, HTML

Details
Dynamics are dynamic (run-time) values that an application associates with an output object or data object. Knowledgeable users will recognize a dynamic as what is inserted into a PROC TEMPLATE DYNAMIC reference when a template and data object are paired during rendering. This example demonstrates how to print a table that shows what dynamics are stored in a specific output object.

Create the ODS document Tab. The ODS DOCUMENT statement opens the DOCUMENT destination. The NAME= option assigns the name Tab to the ODS
document that contains the information from the PROC TABULATE program. Note that by default Tab is created in the Work library. Assign a libref to create Tab in a permanent library. Close the DOCUMENT destination. If the DOCUMENT destination is not closed, no DOCUMENT procedure output can be viewed.

```sas
ods document name=tab(write);
proc tabulate data=sashelp.class;
  class sex age;
  var height;
  table sex, age*height*(mean median);
run;
ods document close;
```

**List the entries of the opened ODS document Tab.** The DOC statement with the NAME= option specifies the current document, Work.Tab. The LIST statement with the LEVELS=ALL option lists detailed information about all levels of the document Work.Tab.

```sas
proc document;
  doc name=tab;
  list / levels=all;
quit;
```

**Output 6.12** Listing of the Tab Document

<table>
<thead>
<tr>
<th>Listing of: Work.Tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order by: Insertion</td>
</tr>
<tr>
<td>Number of levels: All</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\Tabulate#1         | Dir   |
| 2   | 
\Tabulate#1\Report#1| Dir   |
| 3   | 
\Tabulate#1\Report#1\Table#1 | Table |

Copy the path from the listing and paste it into the OBDYNAM statement to display the dynamic variables: The DIR statement changes the current directory to Tabulate#1\Report#1. The path Tabulate#1\Report#1 was obtained from the listing of the Work.Tab document **Output 6.21 on page 184**. The OBDYNAM statement creates a table that displays the dynamics for the Table#1 output object.

```sas
proc document name=tab;
  obdynam \Tabulate#1\Report#1\Table#1;
run;
quit;
```

**Alternatively, you can set the directory and then specify only the last level of the path.** This next step is equivalent to the preceding step:

```sas
proc document name=tab;
  dir \Tabulate#1\Report#1;
  obdynam Table#1;
quit;
```
Example 6: Output Dynamics to a Data Set, Modify the Dynamics, and Use Them in Subsequent Graphs

Features:

- PROC DOCUMENT statement option
  - NAME=
- REPLY statement option
  - DYNAMDATA=
- LIST statement
- OBDYAM statement
- Procedure output
  - PROC DOCUMENT
  - PROC REG

ODS destinations:

- DOCUMENT, OUTPUT, HTML, LISTING

Details

The following example creates a data set that contains the dynamic variables for a fit plot graph created by PROC REG. The dynamic variables are then modified and saved in a new data set. The new data set is used to replay the fit plot.

Create the ODS document Mydoc. The ODS DOCUMENT statement opens the DOCUMENT destination. The NAME= option assigns the name Mydoc to the ODS document that contains the information from the PROC REG procedure. Close the DOCUMENT destination. If the DOCUMENT destination is not closed, no DOCUMENT procedure output can be viewed.
ods graphics on;
ods document name=Mydoc (write);
proc reg data=sashelp.class;
ods select fitplot;
  model weight=height;
quit;
ods document close;

List the entries of the opened ODS document Mydoc. The LIST statement with the LEVELS=ALL option lists detailed information about all levels of the document Work.Mydoc.

proc document name=Mydoc;
  list / levels=all;
quit;

Output 6.14  Listing of the ODS Document Mydoc

<table>
<thead>
<tr>
<th>Listing of: Work.Mydoc\</th>
<th>Order by: Insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of levels: All</td>
<td></td>
</tr>
<tr>
<td>Obs</td>
<td>Path</td>
</tr>
<tr>
<td>1</td>
<td>\Reg#1</td>
</tr>
<tr>
<td>2</td>
<td>\Reg#1\MODEL1#1</td>
</tr>
<tr>
<td>3</td>
<td>\Reg#1\MODEL1#1\ObswiseStats#1</td>
</tr>
<tr>
<td>4</td>
<td>\Reg#1\MODEL1#1\ObswiseStats#1\Weight#1</td>
</tr>
<tr>
<td>5</td>
<td>\Reg#1\MODEL1#1\ObswiseStats#1\Weight#1\FitPlot#1</td>
</tr>
</tbody>
</table>

Display the dynamic variables for the fit plot and store them in a SAS data set. Both the data object and the dynamic variables are stored in the ODS document. The OBDDYNAM statement displays the dynamics, and the ODS OUTPUT statement stores them in a data set named Dynamics. The path specified in the OBDDYNAM statement was copied from the listing produced by the LIST statement.

proc document name=Mydoc;
  ods output dynamics=Dynamics;
obddynam \Reg#1\MODEL1#1\ObswiseStats#1\Weight#1\FitPlot#1;
quit;

View the dynamic variables. You can use PROC PRINT to print the data set Dynamics, which contains the dynamic variables.

proc print noobs data=Dynamics;
run;

The Dynamics data set shows that the values of the dynamic variables are captured in two variables, cValue1 and nValue1. When the dynamic variable is numeric, nValue1 captures its numeric value, and cValue1 captures its formatted value. When the dynamic variable is character, nValue1 is missing, and cValue1 captures its formatted value.
Replay the fit plot using the dynamic variables from the Dynamics data set. The REPLAY statement with the DYNAMDATA= option specified replays the fit plot and uses the dynamic variables from the Dynamics data set. Because no modifications were made, the replayed fit plot is the same as the original replayed fit plot. See Output 6.25 on page 188.

```plaintext
proc document name=Mydoc;
   replay \Reg#1\MODEL1#1\ObswiseStats#1\Weight#1\FitPlot#1 /
       dynamdata=Dynamics;
quit;
```
Modify the values of the dynamic variables and store them in a data set named Dynamics2. You can modify the values of the dynamic variables and store them in a new data set.

```sas
data Dynamics2;
  set Dynamics;
  if label1 = '_SHOWNPARM'
    then nvalue1 = 0;
  if label1 = '_SHOWAIC'
    then nvalue1 = 1;
  if label1 = '_SHOWSBC'
    then nvalue1 = 1;
  if label1 = '_SHOWDEPMEAN'
    then nvalue1 = 1;
  if label1 in ('_RSQUARE'
                '_ADJRSQ')
    then nvalue1 = round(nvalue1, 0.001);
run;
```

Replay the graph and explicitly specify that the values of the dynamic variables come from the ODS output data set Dynamics2. By default, the dynamic variables used are the ones that are stored in the ODS document. The DYNAMDATA= option specifies that the dynamic variables come from the data set Dynamics2.

```sas
proc document name=Mydoc;
  replay \Reg#1\MODEL1#1\ObswiseStats#1\Weight#1\FitPlot#1 /
```
The following fit plot uses new dynamic variables. Notice that the values in the statistics table have changed.

Output 6.17  Fit Plot Using Modified Dynamic Variable Values
Overview: ODSLIST Procedure

What Does the ODSLIST Procedure Do?

The ODSLIST procedure is used to create bulleted lists. With PROC ODSLIST, you can create text templates for lists that can be customized and nested an infinite number of times. You can use style attributes and formats to customize your content and WHERE expressions to specify list item content. With PROC ODSLIST, you can use the DATA= option to bind your data to a template without using a DATA step.

PROC ODSLIST can be used with any output destination. However, PROC ODSLIST is essential for creating content for the ODS destination for PowerPoint and e-books. See “ODS EPUB Statement” in SAS Output Delivery System: User's

Syntax: ODSLIST Procedure

PROC ODSLIST <CONTENTS=text-string><DATA=SAS-data-set>
<NAME=template-name>
    <PAGEBREAK=NO | YES> <PRINT> <STORE=template-store>;
CELLSTYLE expression-1 AS <style-element-name><[style-attribute-specification(s)]>
    < expression-n AS <style-element-name><[style-attribute-specification(s)]>;
DYNAMIC variable-name-1 <<=value–1'expression-1'>
    < variable-name-2 <<=value-2'expression-2'...;>
MVAR variable-name-1 <<=value–1'expression-1'>
    < variable-name-2 <<=value-2'expression-2'...;>
NMVAR variable-name-1 <<=value–1'expression-1'>
    < variable-name-2 <<=value-2'expression-2'...;>
ITEM <expression> / <FORMAT=format-name> <STYLE=style-override>
    <VALUE=integer-value>;
P expression / <FORMAT=format-name> <STYLE=style-override>;
LIST / <START=integer-value><STYLE=style-override> ;
    ITEM <expression> /
        <FORMAT=format-name> <STYLE=style-override>
        <VALUE=integer-value>;
END;
END;
TRANSLATE expression-1 INTO expression-2 < , expression-n INTO expression-m;>

PROC ODSLIST Statement

Creates an item list.

Syntax

PROC ODSLIST <CONTENTS=text-string><DATA=SAS-data-set>
<NAME=template-name>
    <PAGEBREAK=NO | YES><PRINT> <STORE=template-store>;

Optional Arguments

CONTENTS="text-string"
specifies the title for the table of contents. The CONTENTS= option overrides the generated table of contents text.
Tip *text-string* is the text that can be seen in the PDF table of contents and in the Results Window folder descriptions.

**DATA=SAS-data-set**
specifies a SAS data set.

Tip No output is produced if you specify the NAME= option without either the DATA= option or the PRINT option. Without the DATA= or PRINT options, ODS creates the template but does not render the output.

**NAME=template-name**
specifies that the content of the procedure should be saved as a template with the specified name.

Tip No output is produced if you specify the NAME= option without either the DATA= option or the PRINT option. Without the DATA= or PRINT options, ODS creates the template but does not render the output.

**PAGEBREAK=NO | YES**
specifies whether the procedure should generate a page break.

**NO**
specifies that no page break is generated.

Alias OFF

**YES**
specifies that a page break is generated.

Alias ON

Default NO

**PRINT**
specifies that the template is printed as well as stored.

Restriction The PRINT option is needed only if the NAME= option is specified without the DATA= option.

Tip No output is produced if you specify the NAME= option without either the DATA= option or the PRINT option. Without the DATA= or PRINT options, ODS creates the template but does not render the output.

**STORE=template-store**
specifies the template store where the compiled template is placed. If you do not specify a name or template store, ODS stores the template in the first writable template store in the ODS path. By default, this template store is Sasuser.Templat.

If you do specify a name and template store, but do not have a libref specified, then the template is stored in the Work directory. For example, the statement proc odslist name=mylist store=mystore; results in a template store named Work.myystore.

Restriction The STORE= option can only be specified if the NAME= option is specified.
END Statement

Ends item and list blocks.

Syntax

END;

CELLSTYLE AS Statement

For tables, sets the style element of the cells in the table or column according to the values of the variables. For text, sets the style attributes of the list items or paragraphs. Use this statement to set the presentation characteristics (such as foreground color and font face) of individual cells or text.

Restriction: The CELLSTYLE AS statement can be used only within a column template, an ODS list, an ODS textblock, or a table template.

Syntax

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 list item, paragraph, or table cell.

If expression resolves to TRUE (a nonzero 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 string, 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 about SAS functions, see *SAS Functions and CALL Routines: Reference.*

**built-in variable**
is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

- **_COLUMN_**
  is a column number. Column numbering begins with 1.
  
  **Alias**
  _COL_
  
  **Example**
  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

- **_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 CELLSSTYLE 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.
  
  **Example**
  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

- **_ROW_**
  is a row number. Row numbering begins with 1.
  
  **Example**
  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

- **_STYLE_**
  is a style element name.
  
  **See**
  For a table of style element names, see Chapter 20, “Style Elements,” on page 797.
  
  **Example**
  “Example 6: Creating Master Templates” on page 583

- **_VAL_**
  is the data value of a cell.
  
  **Tip**
  Use _VAL_ to represent the value of the current column.
  
  **Example**
  “Example 6: Creating Master Templates” on page 583

**comparison-operator**
compares a variable with a value or with another variable.
The following table lists the comparison operators:

### Table 7.1 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ¬= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>

**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. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**See**

You can use any expression that can be used in the `WHERE=` data set option. For information about expressions that you can use in the `WHERE` data set option, see the `WHERE` data set option in SAS Data Set Options: Reference and the section on `WHERE-Expression Processing` in SAS Language Reference: Concepts.

**Example**

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

**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 about the style attributes that you can set in a table template, see "About Style Attributes" on page 459.

Optional Argument

**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 by using the “DEFINE STYLE Statement” on page 448. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.
The following style elements are most likely to be used with the CELLSTYLE AS statement:

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataEmphasisFixed
- DataStrong
- DataStrongFixed
- ListItem
- ListItem2
- Paragraph

The style element provides the basis for displaying the cell. Additional style attributes modify the display.

**Default** Data

**See** Chapter 14, “TEMPLATE Procedure,” on page 425

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

---

**DYNAMIC Statement**

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

**Restriction:** The DYNAMIC statement can be used only in the template of an ODS textblock, ODS list, 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.

**Syntax**

```plaintext
DYNAMIC variable-name-1 <=value-1<text-1>>
< variable-name-2 <=value-2<text-2'...>>;
```

**Required Argument**

`variable-name`

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.
Optional Arguments

*value*
sets the value of the variable.

*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.

---

### ITEM Statement

Specifies an item to be added to the item list. An ITEM statement that does not specify an expression begins an item block.

Restriction: ITEM statements that do not specify an *expression* must end with an END statement.

#### Syntax

```
ITEM <expression> / <FORMAT=format-name> <STYLE=style-override> <VALUE=integer-value>;
```

#### Optional Arguments

*expression* is an expression that specifies the content of an item.

*expression* has this form:

```
expression-1 < expression-n>
```

*expression* is an arithmetic or logical sequence of operators and operands. An operator is a symbol that requests a logical operation, a string, or an arithmetic calculation.

An operand is one of the following:

*constant* is a fixed value, such as the name of a column, a text string, or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

*function* specifies a SAS function. For information about SAS functions, see *SAS Functions and CALL Routines: Reference*.

Restriction: ITEM statements that do not specify an *expression* must end with an END statement.

*FORMAT=format-name* specifies a default format for the value in each table cell or list item. You can use any SAS or user-defined format.
STYLE=<style-element-name>[style-attribute-name=style-attribute-value]<…
style-attribute-name=style-attribute-value>

specifies the style element to use for the specified item. For example, the
following statement specifies that the text color for the item is red:

```
item 'first item' / style=[color=red];
```

Note: You can use braces ({} and {}) instead of square brackets ([ and ]).

**style-element-name**

is the name of a style element that is part of a style template that is registered
with the Output Delivery System. SAS provides some style templates. You
can create your own style templates with PROC TEMPLATE.

See Chapter 14, “TEMPLATE Procedure,” on page 425 for information about
PROC TEMPLATE and the default style templates.

For a list of style elements, see Chapter 20, “Style Elements,” on page 797.

**style-attribute-name**

specifies the attribute to change.

See For information about style attributes and their values, see Chapter 21,
“Style Attributes,” on page 827.

**style-attribute-value**

specifies a value for the attribute. Each attribute has a different set of valid
values.

See For information about style attributes and their values, see Chapter 21,
“Style Attributes,” on page 827.

**VALUE=integer-value**

starts a numbered list from the specified integer-value. If another VALUE= option
is specified in subsequent statements, numbering begins again at that point.

Restriction When VALUE= is specified, you must also specify a numbering list
type with the LISTSTYLETYPE= style attribute. For example,
LISTSTYLETYPE= "decimal_leading_zero" or
LISTSTYLETYPE= "decimal".

Tip You can change the bullet type of a list item by specifying the
"LISTSTYLETYPE=bullet-type" style attribute with the STYLE= option.

Example “Example 2: Creating Nested Lists” on page 216

---

**LIST Statement**

Creates a list.

Restrictions: The LIST statement must be used within an item block.
LIST statements begin a LIST statement block, and must end with an END statement.
Example: "Example 4: Comparing Lists Created with PROC ODSLIST and PROC ODSTEXT" on page 223

Syntax

```
LIST / <START=integer-value><STYLE=style-override> ;
```

Optional Arguments

**START=integer-value**

starts a numbered list from the specified integer-value. If another START= option is specified in subsequent statements, the numbering begins again at that point.

**Restrictions**

When START= is specified, you must also specify a numbering list type with the LISTSTYLETYPE= style attribute. For example, LISTSTYLETYPE= "decimal_leading_zero" or LISTSTYLETYPE= "decimal" are numbering list types.

The START= option is valid for the PDF, POWERPOINT, HTML, and RTF destinations.

**Tip**

You can change the bullet type of a list item by specifying the "LISTSTYLETYPE=bullet-type" style attribute with the STYLE= option.

**Examples**

The following example creates a list that begins numbering items with the number seven. For HTML5 output, if you want to start a numbered list from a specific number with the START= option, then specify LISTSTYLETYPE= on the LIST statement, as this example does.

```sas
proc odslist;
  item;
    p 'This is a numbered list';
    list / style={liststyletype="decimal_leading_zero"} start=7;
    item "Seven";
    item "Eight";
  end;
end;
run;
```

The following example creates a list that assigns a specific number to each item.

```sas
proc odslist;
  item;
    p 'This is a numbered list';
    list / style={liststyletype="decimal_leading_zero"};
    item "Two" / value=2;
    item "Four" / value=4;
    item "Six" / value=6
  end;
end;
run;
```
**STYLE**=<b>style-element-name</b> [<b>style-attribute-name=style-attribute-value</b>… <b>style-attribute-name=style-attribute-value</b>]

specifies the style element to use for the specified item in the list. For example, the following statement specifies that the text color for the item is red:

item 'first item' / style=[color=red];

---

**Note:** You can use braces ({ and }) instead of square brackets ([ and ]).

---

**style-element-name**

is the name of a style element that is part of a style template that is registered with the Output Delivery System. SAS provides some style templates. You can create your own style templates with PROC TEMPLATE.

See Chapter 14, “TEMPLATE Procedure,” on page 425 for information about creating style templates with PROC TEMPLATE.

For a list of style elements, see Chapter 20, “Style Elements,” on page 797.

---

**style-attribute-name**

specifies the attribute to change.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

---

**style-attribute-value**

specifies a value for the attribute. Each attribute has a different set of valid values.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

---

**MVAR Statement**

Defines a symbol that references a macro variable. ODS uses 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.

**Restriction:** When replaying an ODS document with PROC DOCUMENT, values created by the MVAR statement must be re-created in the same session that is replaying the document.

**Tip:** You can use the MVAR statement in the template of an ODS list, ODS textblock, 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.

**See:** “Example 3: Creating a New Table Template” on page 563 and “Example 1: Creating a Stand-Alone Style” on page 459

---

**Syntax**

MVAR variable-name-1 <='value-1' <'text-1'>
    < variable-name-2 <='value-2'> <'text-2'...>>;
Required Argument

\textit{variable-name} 

names a macro variable to reference in the template. ODS uses 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 \texttt{SYSDATE9} in a template, declare it in an \texttt{MVAR} statement and reference it as \texttt{SYSDATE9}, without an ampersand, in the \texttt{PROC TEMPLATE} or \texttt{PROC ODS\_TABLE} 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.

Optional Arguments

\textit{value} 

sets the default variable value.

\textit{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 \texttt{SOURCE} statement, whereas SAS comments do not.

\section*{NMVAR Statement}

Defines a symbol that references a macro variable. ODS converts 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.

Restriction: The NMVAR statement can be used only in the template of an ODS list, ODS textblock, 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.

See: “Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

\section*{Syntax}

\texttt{NMVAR variable-name-1 \textasciitilde=value-1 \textasciitilde=text-1\>
< variable-name-2 \textasciitilde=value-2 \textasciitilde=text-2…\>;

Required Argument

\textit{variable-name} 

names a macro variable to reference in the template. ODS converts 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.
Optional Arguments

_value_
sets the value of the variable.

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.

---

**P Statement**

Specifies a paragraph. Multiple P statements are allowed within an item block.

Restriction: In the ODSLIST Procedure, the P statement can be specified only within an ITEM block. In the ODSTEXT procedure, the P statement can be used at the top level of PROC ODSTEXT as well as within list items.

---

**Syntax**

\[ P \ expression \ / \ <\text{FORMAT}=\text{format-name}> <\text{STYLE}=\text{style-override}> ; \]

**Required Argument**

_expression_

is an expression that specifies the content of an item or paragraph. _expression_ has this form:

\[ expression-1 < expression-n \]

_expression_

is an arithmetic or logical sequence of operators and operands. An operator is a symbol that requests 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, text string, or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

For example, the following code creates a paragraph and an item list with PROC ODSTEXT:

```sas
proc odstext data=sashelp.class;
  p "My name is " || name;
  list;
    item;
      p "My age is " || put(age, 2.);
      p "My weight is " || put(weight, 3.);
    end;
  end;
run;
```

The following code creates an item list with PROC ODSLIST:

```sas
proc odslist data=sashelp.class;
  item;
```
p "My name is " || name;
list;
item;
p "My age is " || put(age, 2.);
p "My weight is " || put(weight, 3.);
end;
end;
end;
run;

function

specifies a SAS function. For information about SAS functions, see SAS Functions and CALL Routines: Reference.

Restriction
ITEM statements that do not specify an expression must end with an END statement.

Optional Arguments

**FORMAT=** *format-name*

specifies a default format for the value in each paragraph. You can use any SAS or user-defined format.

**STYLE=** `<style-element-name >[style-attribute-name=style-attribute-value]<...style-attribute-name=style-attribute-value>`

specifies the style element to use for the items in the list or paragraph.

Note: You can use braces ({ and }) instead of square brackets ([ and ]).

**style-element-name**

is the name of a style element that is part of a style template that is registered with the Output Delivery System. SAS provides some style templates. You can create your own style templates with PROC TEMPLATE.

See Chapter 14, "TEMPLATE Procedure," on page 425 for information about PROC TEMPLATE and the default style templates.

For a list of style elements, see Chapter 20, "Style Elements," on page 797.

**style-attribute-name**

specifies the attribute to change.

See For information about style attributes and their values, see Chapter 21, "Style Attributes," on page 827.

**style-attribute-value**

specifies a value for the attribute. Each attribute has a different set of valid values.

See For information about style attributes and their values, see Chapter 21, "Style Attributes," on page 827.

**Examples**

For example, the following statement specifies that the text color of the paragraph is red:

p 'text block' / style=[color=red];
For example, the following statement specifies that the text color for the item is red:

```
item 'first item' / style=[color=red];
```

## TRANSLATE INTO Statement

Translates the specified numeric values to other values.

### Restrictions:
- The TRANSLATE INTO statement can be used only in a column template, an ODS list, an ODS textblock, or a table template.
- 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.

### Syntax

```
TRANSLATE expression-1 INTO expression-2 <, expression-n INTO expression-m>
```

### Required Arguments

**expression-1**

is an expression that is evaluated for each list item, paragraph, table, or column cell that contains a numeric variable.

If `expression-1` resolves to TRUE (a nonzero 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 about SAS functions, see *SAS Functions and CALL Routines: Reference*.

  - `built-in variable` is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

    - `_COLUMN_` is a column number. Column numbering begins with 1.
Alias: \_COL\_

Example: “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

\_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.

Example: “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

\_ROW\_ is a row number. Row numbering begins with 1.

Example: “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

\_STYLE\_ is a style element name.

Example: “Example 6: Creating Master Templates” on page 583

\_VAL\_ is the data value of a cell.

Tip: Use \_VAL\_ to represent the value of the current column.

Example: “Example 6: Creating Master Templates” on page 583

comparison-operator compares a variable with a value or with another variable. The following table lists the comparison operators:

**Table 7.2 Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or =~ or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>
### Symbol

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>Equal to one or more from a list of values</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

### See

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in *SAS Data Set Options: Reference* and the section on WHERE-Expression Processing in *SAS Language Reference: Concepts*.

### Example

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

---

**expression-2**

is an expression that specifies the value to use in the list, paragraph, or 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 about SAS functions, see *SAS Functions and CALL Routines: Reference*.

- **Built-in variable**
  
a special type 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.

  - **_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

Example  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_ROW_
is a row number. Row numbering begins with 1.

Example  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_STYLE_
is a style element name.

See  For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

Example  “Example 6: Creating Master Templates” on page 583

.VAL_
is the data value of a cell.

Tip  Use _VAL_ to represent the value of the current column.

Example  “Example 6: Creating Master Templates” on page 583

comparison-operator
compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 7.3  Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or <del>= or =</del> or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

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 or 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 formatted field, starting at the left. To right-justify the value, use the JUSTIFY=ON attribute.

See

“JUSTIFY<=ON | OFF | variable>;” on page 602 column attribute

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

Example

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

Usage: ODSLIST Procedure

Using the ODSLIST Procedure

PROC ODSLIST uses many of the same statements as PROC TEMPLATE to create and customize your list templates. The CELLSTYLE AS, DYNAMIC, MVAR, NMVAR, and TRANSLATE INTO statements are all valid statements within PROC ODSLIST that can be used to customize your list templates.

Changing the Bullet Type for Lists

By default, lists created by the ODSLIST procedure are displayed as an unordered list with a disc as the bullet type. In markup family output, you can change the bullet type for lists by using the following two steps:

1. Specify the LISTSTYLETYPE= style attribute with the desired bullet type in a style override. The following shows the general syntax for specifying the LISTSTYLETYPE= attribute:

   STYLE={ LISTSTYLETYPE="bullet-type" <more-style-attributes>}

2. Specify the style override in an ITEM or LIST statement. The following shows the general syntax for specifying a style override in the LIST or ITEM statements:

   LIST / STYLE={ LISTSTYLETYPE="bullet-type" <more-style-attributes>}
   ITEM / STYLE={ LISTSTYLETYPE="bullet-type" <more-style-attributes>}

Note:
For HTML5 output, if you want to start a numbered list from a specific number with the START= option, then specify LISTSTYLETYPE= in the LIST statement.

**IMPORTANT**

The LISTSTYLETYPE= style attribute is valid in the Markup family, Excel, and PowerPoint destinations. The valid bullet-type values differ for each destination. For more information about valid bullet types, see the LISTSTYLETYPE= Style Attribute on page 877.

When you specify LISTSTYLETYPE= in the LIST statement, the bullets for all of the following items are changed. If you specify LISTSTYLETYPE= in an ITEM statement, only the bullet for that ITEM statement is changed.

The following example creates a numbered list. Because the style override is specified in the LIST statement, all of the following items have the same numbering scheme.

```plaintext
title;
proc odslist;
    item / style={liststyletype="none" fontsize=12pt};
    p 'Follow these steps to change the bullet type for lists:'
       / style={fontsize=12pt};
    list / style={liststyletype="decimal" fontsize=12pt};
       item "Decide what bullet type to display."
       item "Add a '/' after the LIST statement (if not already present)."
       item "Specify the LISTSTYLETYPE= attribute with the STYLE= option.";
    end;
end;
run;
```

**Output 7.1  Changing the Bullet Type for All Items**

Follow these steps to change the bullet type for lists:

1. Decide what bullet type to display.
2. Add a '/' after the LIST statement (if not already present).
3. Specify the LISTSTYLETYPE= attribute with the STYLE= option.

You can also change the bullet type for each individual item by specifying a style override in each ITEM statement.

```plaintext
title 'You Can Change the Bullet Type for Each Individual Item' ;
proc odslist;
    item "List items can be a square" / style={liststyletype="square" fontsize=12pt};
    item "Or a circle" / style={liststyletype="circle" fontsize=12pt};
    item "Or a disc" / style={liststyletype="disc" fontsize=12pt};
    item "Or items can have no bullet" / style={liststyletype="none" fontsize=12pt};
end;
run;
```
Changing the Bullet Type for Individual Items

The following are browser-supported bullet types. However, many browsers support other common types and the support is browser dependent. If an unsupported bullet type is specified, the bullet’s display is browser dependent.

Table 7.4 Valid Bullet Types

<table>
<thead>
<tr>
<th>Bullet Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ordered Lists</strong></td>
<td></td>
</tr>
<tr>
<td>decimal</td>
<td>The list items are ordered with numbers (default).</td>
</tr>
<tr>
<td>upper_latin</td>
<td>The list items are ordered with uppercase letters.</td>
</tr>
<tr>
<td>lower_latin</td>
<td>The list items are ordered with lowercase letters.</td>
</tr>
<tr>
<td>upper_roman</td>
<td>The list items are ordered with uppercase roman numerals.</td>
</tr>
<tr>
<td>lower_roman</td>
<td>The list items are ordered with lowercase roman numerals.</td>
</tr>
<tr>
<td><strong>Unordered Lists</strong></td>
<td></td>
</tr>
<tr>
<td>disc</td>
<td>The list items are marked with filled circles.</td>
</tr>
<tr>
<td>circle</td>
<td>The list items are marked with circles.</td>
</tr>
<tr>
<td>square</td>
<td>The list items are marked with squares.</td>
</tr>
<tr>
<td>none</td>
<td>The list items are not marked.</td>
</tr>
</tbody>
</table>
Examples: ODSLIST Procedure

Example 1: Using Item Statements

Features:
- CELLSTYLE AS statement
- PROC ODSLIST statement
  - NAME= option
  - PRINT option
  - STORE= option
- ITEM statement
  - STYLE= option
- ODS HTML CLOSE statement
- ODS POWERPOINT statement
- OPTIONS statement
- TITLE statement

Details

The following program creates a list for a Microsoft PowerPoint slide. The CELLSTYLE AS statement formats the contents of each item.

Program

```sas
ods html close;
options nodate;
title 'Using PROC ODSLIST ITEM Statements';

ods powerpoint file="DefaultStyle.ppt";
ods powerpoint(2) file="PowerpointdarkStyle.ppt" style=powerpointdark;

proc odslist name=Slides store=sasuser.Myexampleslides print;
  cellstyle 1 as {fontsize=1cm color=purple fontweight=bold};
  item 'Fraud';
  item 'Customer Intelligence';
  item 'Social Media';
  item 'Data Mining';
  item 'High-Performance Computing';
  item 'Risk';
  item 'Data Management';
run;

ods _all_ close;
```

Program Description

**Close the HTML destination and specify the SAS system options.** The HTML destination is open by default in SAS Windowing environment. If you are not creating HTML output, then close the destination to save system resources.

```sas
ods html close;
options nodate;
title 'Using PROC ODSLIST ITEM Statements';
```

**Open two instances of the ODS destination for PowerPoint.** The ODS POWERPOINT statement creates output formatted for the ODS destination for PowerPoint.

```sas
ods powerpoint file="DefaultStyle.ppt";
ods powerpoint(2) file="PowerpointdarkStyle.ppt" style=powerpointdark;
```

**Begin the ODSLIST procedure and create a template store and name for the template.** The NAME= option gives the template a name and the STORE= option specifies the template store where the template is stored. If the specified template store does not exist, then ODS creates it. Because there is no DATA= option specified, you must use the PRINT option to render the output. Otherwise, the template is created but no output is rendered.

```sas
proc odslist name=Slides store=sasuser.Myexampleslides print;
```

**Create content for the ODS destination for PowerPoint.** The ITEM statements create each item in the list. The CELLSTYLE AS statement applies the style attributes to each item.

```sas
cellstyle 1 as {fontsize=1cm color=purple fontweight=bold};
item 'Fraud';
item 'Customer Intelligence';
item 'Social Media';
item 'Data Mining';
item 'High-Performance Computing';
item 'Risk';
item 'Data Management';
run;
```

**Close the open destinations.**

```sas
ods _all_ close;
```
Example 2: Creating Nested Lists

Features:

- ITEM statement
  - STYLE= option
  - VALUE option
- LIST statement
- P statement
- PROC ODSLIST statement
  - NAME= option
  - PRINT option

- Fraud
- Customer Intelligence
- Social Media
- Data Mining
- High-Performance Computing
- Risk
- Data Management
Details

PROC ODSLIST enables you to easily nest lists and add text. The following example creates a nested list inside the first item of the list. You can use the P statement to add text that is not in a list.

Program

```sas
ods html close;
options nodate;
title 'Creating Nested Lists';
ods powerpoint file="NestedList.ppt" style=powerpointdark;
proc odslist name=nested store=sasuser.Myexampleslides print;

   item;
      p ' Fraud' ;
      list;
         item 'Consumer Fraud';
         item 'Business Fraud';
      end;
   end;
   item ' Customer Intelligence';
   item ' Social Media' ;
   item ' Data Mining';
   item ' High-Performance Computing';
   item ' Risk';
   item ' Data Management';
run;
ods _all_ close;
```

Program Description

**Close the HTML destination and set the SAS system options.** The HTML destination is open by default in SAS Windowing environment. If you are not creating HTML output, then close the destination to save system resources.

```sas
ods html close;
options nodate;
title 'Creating Nested Lists';
```

**Open the ODS destination for PowerPoint.** The ODS POWERPOINT statement creates output formatted for the ODS destination for PowerPoint.

```sas
ods powerpoint file="NestedList.ppt" style=powerpointdark;
```
Begin the ODSLIST procedure and create a template store and name for the template. The NAME= option gives the template a name and the STORE= option specifies the template store to put the template in. If the specified template store does not exist, then ODS creates it. Because there is no DATA= option specified, you must use the PRINT option to render the output. Otherwise, the template is created, but no output is rendered.

```
proc odslist name=nested store=sasuser.Myexampleslides print;
```

**Create a nested list.** The first ITEM statement creates the list item that contains the nested list. The P statement specifies the paragraph that contains the nested list. The LIST statement begins the list, and the ITEM statements create the list items.

```
item;
  p ' Fraud' ;
  list;
    item 'Consumer Fraud';
    item 'Business Fraud';
  end;
end;
```

**Create the remaining list items.** The ITEM statements that are not nested create the remaining list items.

```
item ' Customer Intelligence';
item ' Social Media' ;
item ' Data Mining';
item ' High-Performance Computing';
item ' Risk';
item ' Data Management';
run;
```

**Close the open destinations.**

```
ods _all_ close;
```
Example 3: Customizing Lists

Features:
- LIST statement
- P statement
- PROC ODSLIST statement
  - NAME= option
  - PRINT option
  - STORE= option
- ITEM statement
  - STYLE= option
- ODS ESCAPECHAR statement
- FOOTNOTE statement
- ODS HTML CLOSE statement
- ODS POWERPOINT statement
- OPTIONS statement
- TITLE statement

Details
You can customize any list created with PROC ODSLIST. You can use the STYLE= option on the ITEM, P, or LIST statement. You can specify customizations for the entire list with the STYLE= option in the LIST statement. You can also make specific changes to each item or text with the STYLE= option on the ITEM or P statement.
Program

ods html close;
options nodate;
ods escapechar = "^";
title 'Customizing Lists';
footnote "Marketing Data "{nbspace 50} www.sas.com ";

ods powerpoint file="a.ppt" style=powerpointdark;
proc odslist name=nested store=sasuser.Myexampleslides print;

item;
  p ' Topics For This Week';
  list / style={liststyletype=decimal fontweight=bold fontsize=1cm color=blue};
  item / style={[fontweight=bold fontsize=1cm color=red] value=1; 
    p ' Fraud' ;
    list / style={liststyletype=decimal color=purple};
    item 'Consumer Fraud '^super January-June'^{sub (* Allen and sons)}}' / value=1;
    item 'Business Fraud';
  end;
end;
item ' Customer Intelligence' ;
item ' Social Media';
item ' Data Mining';
item ' High-Performance Computing';
item ' Risk '{style [font_style=italic color=purple] ^{sub (* Allen and sons)}}';
item ' Data Management';
end;
end;
run;

title;
proc odslist;

item;
  p ' Topics For Today';
  list / style={liststyletype=decimal };
  item ' Customer Intelligence' / style={[fontweight=bold fontsize=1cm color=purple] value=1};
  item ' Social Media' / style={[fontweight=bold fontsize=1cm ]};
  item ' Data Mining' / style={[fontweight=bold fontsize=1cm ]};
end;
end;
run;

ods _all_ close;

Program Description

Close the HTML destination and specify the SAS system options. The HTML destination is open by default in SAS Windowing environment. If you are not creating HTML output, then close the destination to save system resources.
Specify titles and footnotes and define a representative character to be used in output strings.

ods escapechar = "^";
title 'Customizing Lists';
footnote "Marketing Data \{nbspace 50\} www.sas.com \";

Open the ODS destination for PowerPoint. The ODS POWERPOINT statement creates output formatted for the ODS destination for PowerPoint.

ods powerpoint file="a.ppt" style=powerpointdark;

Begin the ODSLIST procedure and create a template store and name for the template. The NAME= option gives the template a name and the STORE= option specifies the template store to put the template in. If the specified template store does not exist, then ODS creates it. Because there is no DATA= option specified, you must use the PRINT option to render the output. Otherwise, the template is created but no output is rendered.

proc odslist name=nested store=sasuser.Myexampleslides print;

Begin creating content for your output. The ITEM statement without an expression specified begins an ITEM block. The P statement specifies explanatory text for the list. The LIST statement begins the list. LIST statements must be specified within an ITEM block. The STYLE= option that is specified in the LIST statement applies to all of the list items within the list.

item;
p ' Topics For This Week';
list / style={liststyletype=decimal fontweight=bold fontsize=1cm color=blue};

Create a nested list. The LIST statement within the P statement creates a nested list for the first item on the list. The STYLE= option in the LIST statement specifies style attributes to format the text for all of the items in the list. The END statements end the LIST and ITEM blocks. A STYLE= option specified in an ITEM statement overrides the STYLE= option specified in the LIST statement. The color “blue” is applied to all of the items in overall list except the first item, which has a STYLE= option specified. The color “red” is applied to the first item only.

item / style=[fontweight=bold fontsize=1cm color=red] value=1;
p ' Fraud';
list / style={liststyletype=decimal color=purple};
item 'Consumer Fraud ^{super January-June}'; / value=1;
item 'Business Fraud';
end;
end;

Create the remaining list items. The ITEM statements that are not nested create the remaining list items. The customization for each item comes from the STYLE= option specified on the first LIST statement. The STYLE= option on the “Risk” item specifies style attributes to format the text for that item only.

item ' Customer Intelligence' ;
item ' Social Media';
item ' Data Mining';
item ' High-Performance Computing';
item ' Risk ^{style [font_style=italic color=purple] ^{sub (* Allen and sons)}}';
Create a second slide. The second ODSLIST procedure block creates a list on a second slide. The STYLE= option specifies style attributes to format the text.

```
CREATE A SECOND SLIDE. THE SECOND ODSLIST PROCEDURE BLOCK CREATES A LIST ON A SECOND SLIDE. THE STYLE= OPTION SPECIFIES STYLE ATTRIBUTES TO FORMAT THE TEXT.
```

```
title;
proc odslist;
  item;
    p 'Topics For Today';
    list / style={liststyletype=decimal};
    item 'Customer Intelligence' / style=[fontweight=bold fontsize=1cm color=purple] value=1;
    item 'Social Media' / style=[fontweight=bold fontsize=1cm];
    item 'Data Mining' / style=[fontweight=bold fontsize=1cm];
  end;
end;
run;
```

Close the open destinations.

```
ods _all_ close;
```

Slide 1

*Output 7.6 Slide 1*

**Customizing Lists**

- **Topics For This Week**
  1. **Fraud**
     1. Consumer Fraud January-June
     2. Business Fraud
  2. **Customer Intelligence**
  3. **Social Media**
  4. **Data Mining**
  5. **High-Performance Computing**
  6. **Risk** (*Allen and sons*)
  7. **Data Management**
Example 4: Comparing Lists Created with PROC ODSLIST and PROC ODSTEXT

Features:

ODSLIST Procedure
- CELLSTYLE AS statement
- END statement
- ITEM statement
- LIST statement
- P statement:
  - PROC ODSLIST statement
  - TRANSLATE INTO statement

ODSTEXT Procedure
- CELLSTYLE AS statement
- END statement
- ITEM statement
- LIST statement
- P statement
  - PROC ODSTEXT statement
  - TRANSLATE INTO statement

OPTIONS statement
TITLE statement

Details

The following examples create similar output using PROC ODSLIST and PROC ODSTEXT. Both procedures can be used to create lists and paragraphs. However,
there are differences in how the syntax of the two procedures can be specified. The following table shows how the P, ITEM, and LIST statements are specified in each procedure.

<table>
<thead>
<tr>
<th>PROC ODSTEXT</th>
<th>PROC ODSLIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>The P statement can be specified only within an ITEM block.</td>
<td>The P statement can be used at the top level of PROC ODSTEXT as well as within list items.</td>
</tr>
<tr>
<td>The ITEM statement must be used within a LIST block.</td>
<td>The ITEM statement does not have to be specified within a LIST block.</td>
</tr>
<tr>
<td>The LIST statement does not have to be specified within an item block.</td>
<td>The LIST statement must be used within an item block.</td>
</tr>
</tbody>
</table>

There are also slight differences in the appearance of the output created by the two procedures. Examine the output created by the two programs in this example. Notice that the spacing and bullet points are slightly different.

Program: Using PROC ODSLIST

```sas
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";

proc odslist data=sashelp.class;
  item;
    p 'List of Students'' Names, Ages, and Weights' / style=systemtitle;
    p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};
  list;
    cellstyle age<=13 as datastrong{background=green},
    age>=14 as {background=lightpurple};
    item 'Student name is ' || name;
    item 'Age: ' || put(age, 2.);
    item;
    translate age<=13 into 'Weight: N/A';
    p 'Weight: ' || put(weight, 3.);
  end;
end;
run;
```

Program Description

Specify the SAS system options and title.

```sas
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";
```
Begin the ODSLIST procedure, begin an ITEM block, and specify explanatory text. The PROC ODSLIST statement specifies the input data set and begins the procedure. The ITEM statement with no options specified begins an ITEM block. The P statements specify the explanatory text. In the ODSLIST Procedure, the P statement must be specified within an ITEM block.

```
proc odslist data=sashelp.class;
  item;
    p 'List of Students'' Names, Ages, and Weights' / style=systemtitle;
    p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};
end;
run;
```

Begin the list and conditionally set the style attributes. The LIST statement with no options specified begins a list block. The LIST statement must be used within an item block. The CELLSTYLE AS statement sets the style attributes of the list items based on the value of the variable Age. If the value of the variable Age is less than or equal to 13, the background color is green. If the value of the variable Age is greater than or equal to 14, the background color is light purple.

```
list;
  cellstyle age<=13 as datastrong{background=green},
      age>=14 as {background=lightpurple};
end;
```

Specify the items. The ITEM statements specify the content of each item.

```
item 'Student name is ' || name;
item 'Age: ' || put(age, 2.);
```

Conditionally transform the value for the AGE variable. The TRANSLATE INTO statement translates the value of Age based on the conditions specified. For values of the variable AGE that are equal to or less than thirteen, “N/A” is written to the output.

```
item;
  translate age<=13 into 'Weight: N/A';
  p 'Weight: ' || put(weight, 3.);
end;
```

End the LIST and ITEM blocks. The LIST block began with the LIST statement. An END statement ends the LIST block. Both ITEM blocks began with an ITEM statement with no expression specified. An END statement ends each ITEM block.

```
end;
end;
run;
```
Program: PROC ODSTEXT

```sas
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";

proc odstext data=sashelp.class;
  p 'List of Students'' Names, Ages, and Weights' / style=systemtitle;
  p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};
  list;
    cellstyle age<=13 as datastrong{background=green},
        age>=14 as {background=lightpurple};
    item 'Student name is ' || name;
    item 'Age: ' || put(age, 2.);
    item;
    translate age<=13 into 'Weight: N/A';
    p 'Weight: ' || put(weight, 3.);
    end;
end;
run;
```

Program Description

**Specify the SAS system options and title.**

```sas
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";
```

**Begin the ODSTEXT procedure and specify explanatory text for the list.** The PROC ODSLIST statement specifies the input data set and begins the procedure. The P statements specify the text that precedes the list. In the ODSTEXT procedure, the P statement does not have to be specified within an ITEM block.
proc odstext data=sashelp.class;
  p 'List of Students'' Names, Ages, and Weights' / style=systemtitle;
  p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};

Begin the list and conditionally set the style attributes. The LIST statement with no options specified begins a list block. In the ODSTEXT procedure, the LIST statement does not have to be specified within an item block. The CELLSTYLE AS statement sets the style attributes of the list items based on the value of the variable Age. If the value of the variable Age is less than or equal to 13, the background color is green. If the value of the variable Age is greater than or equal to 14, the background color is light purple.

list;
  cellstyle age<=13 as datastrong{background=green},
    age>=14 as {background=lightpurple};

Specify the items. The ITEM statements specify the content of each item.

item 'Student name is ' || name;
item 'Age: ' || put(age, 2.);

Conditionally transform the value for the AGE variable. The TRANSLATE INTO statement translates the value of Age based on the conditions specified. For values of the variable AGE that are equal to or less than thirteen, "N/A" is written to the output. The TRANSLATE INTO statement must be specified within the block of the variable you want to be translated. Otherwise, the variable is translated and the results apply to all variables.

item;
  translate age<=13 into 'Weight: N/A';
  p 'Weight: ' || put(weight, 3.);
end;

End the LIST block. The LIST block began with the LIST statement. An END statement ends the LIST block.

end;
run;
### Output 7.9  Output Created with the PROC ODSTEXT

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is Alice**
  - Age: 13
  - Weight: 96

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is Bob**
  - Age: 13
  - Weight: 96

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is Carol**
  - Age: 13
  - Weight: 96

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is Dave**
  - Age: 13
  - Weight: 96

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is Emma**
  - Age: 13
  - Weight: 96

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is Fred**
  - Age: 13
  - Weight: 96

**List of Students’ Names, Ages, and Weights**

Weights of children younger than fourteen are hidden for legal reasons

- **Student name is George**
  - Age: 13
  - Weight: 96
Overview: ODSTABLE Procedure

What Does the ODSTABLE Procedure Do?

By default, ODS output is formatted according to the various definitions or templates that the procedure or DATA step specify. Table templates describe how tables...
should be constructed. This includes the content and placement of headers and
footers, the content and placement of columns, and style overrides. All SAS
procedures, except PROC PRINT, PROC REPORT, and PROC TABULATE, use
table templates to describe how their tables should look. This means that you can
change the structure of tables that are generated by SAS procedures by using table
templates.

You can create your own new tabular output templates by using the ODSTABLE
procedure. The Output Delivery System then uses these templates to produce
customized tabular output. With the ODSTABLE procedure, you can create table
templates and bind them with the input data set in one statement. You can also
name your templates and store them in a template store.

Concepts: ODSTABLE Procedure

Comparing PROC TEMPLATE and PROC ODSTABLE

The ODSTABLE procedure is a simpler way of producing the same output that you
would expect to get from using the DEFINE TABLE statement in PROC TEMPLATE.
You can use the same statements in an ODSTABLE block as you would use in a
DEFINE TABLE block in PROC TEMPLATE. For example, the table templates
created in both of the following programs are equivalent. The statements between
the DEFINE TABLE statement and the END statement in the PROC TEMPLATE
step are identical to the statements in the PROC ODSTABLE step.

Example Code 8.1  Table Template Created with PROC TEMPLATE

```
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) (grange) (p1) (p5) (p10) (p25) (p50) (p75)
      (p90) (p95) (p99);
    define nobs;
      style={color=orange backgroundcolor=white};
    end;
  end;
run;
```

Example Code 8.2  Table Template Created with PROC ODSTABLE

```
proc odstable name=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) (sumwt) (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=orange backgroundcolor=white};
end;
run;

Table 8.1 Differences between PROC TEMPLATE and PROC ODSTABLE

<table>
<thead>
<tr>
<th>PROC TEMPLATE with DEFINE TABLE Statement</th>
<th>PROC ODSTABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates and modifies table templates.</td>
<td>Creates and modifies table templates.</td>
</tr>
<tr>
<td>Creates and modifies column, header, and footer templates in a single procedure step.</td>
<td>Improves the readability of programs by creating one template type in one procedure step.</td>
</tr>
<tr>
<td>Must use a DATA <em>NULL</em> step to bind the data to the table template.</td>
<td>Creates data-dependent table templates without using the DATA <em>NULL</em> step.</td>
</tr>
<tr>
<td>Does not require an input data set.</td>
<td>No output is produced if you specify the NAME= option without the DATA= option. Without the DATA= option, ODS creates the template but does not render the output.</td>
</tr>
</tbody>
</table>

Syntax: ODSTABLE Procedure

PROC ODSTABLE DATA=data-set-name | NAME=template-name
  <PAGEBREAK= NO | YES >| STORE=template-store>;
  <table-attribute-1 | table-attribute-2>…>;
  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 </ options>;
  statements-and-attributes
END;
### PROC ODSTABLE Statement

Begins a table template.

**Requirement:** You must specify the DATA= option or the NAME= option.

**Syntax**

```bash
PROC ODSTABLE DATA=data-set-name | NAME=template-name <CONTENTS="text-string"> <PAGEBREAK=NO | YES> <STORE=template-store>; <table-attribute-1 < table-attribute-2>...>
```

**Required Arguments**

- **DATA=** `data-set-name`
  
  Specifies the input data.

  **Examples**
  
  - “Example 1: Customizing Columns” on page 268
  
  - “Example 3: Creating and Storing a Customized Table Template” on page 272

- **NAME=** `template-name`
  
  Specifies the name of the template. Specifying a template name with the NAME= option is the same as specifying a template name with the DEFINE TABLE statement in PROC TEMPLATE.

  For example, the following two code blocks are equivalent:

  ```bash
  proc odstable name=mytable;
  run;
  
  proc template;
  define table mytable;
  end;
  run;
  ```

  **Examples**
  
  - “Example 3: Creating and Storing a Customized Table Template” on page 272
Optional Arguments

**CONTENTS=**"text-string"

specifies the title for the table of contents. The CONTENTS= option overrides the generated table of contents text.

Tip  *text-string* is the text that can be seen in the PDF table of contents and in the Results Window folder descriptions.

**PAGEBREAK= NO | YES**

specifies whether the procedure should generate a page break.

NO

specifies that no page break is generated.

Alias  OFF

YES

specifies that a page break is generated.

Alias  ON

Default  YES

**STORE=**template-store

specifies the template store where the compiled template is placed. If you do not specify a name or template store, ODS stores the template in the first writable template store in the ODS path. By default, this template store is Sasuser.Templat.

If you do specify a name and template store, but do not have a libref specified, then the template is stored in the Work directory. For example, the statement  

```
proc odstable name=mylist store=mystore;
```

results in a template store named Work.mystore.

Using the STORE= option is the same as using the STORE= option in the DEFINE statement in PROC TEMPLATE. For example, the following two code blocks are equivalent:

```
proc odstable name=mytable store=sasuser.mystore(update);
run;
```

```
proc template;
   define table mytable / store=sasuser.mystore(update);
end;
run;
```

Restriction  The STORE= option can only be used if the NAME= option is specified.

Examples  "Example 3: Creating and Storing a Customized Table Template" on page 272

"Example 4: Using PROC TEMPLATE and PROC ODSTABLE " on page 274
Table Attributes

This section lists all the attributes that you can use in a table template. Table attributes are used to customize the attributes of a table. You can specify multiple table attributes together or separately. For example, you can specify the following table attributes together:

order_data=yes use_format_defaults=yes print_headers=off;

or separately:

order_data=yes;
use_format_defaults=yes;
print_headers=off;

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>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td><em>ON</em></td>
<td><em>OFF</em></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><em>YES</em></td>
<td><em>NO</em></td>
</tr>
</tbody>
</table>

Table 8.3 Table Attributes

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
<td>BALANCE on page 624</td>
<td>LISTING, printer family, and RTF</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Specify whether to center each data panel independently if the entire table does not fit in one data panel</td>
<td>CENTER on page 624</td>
<td>LISTING, printer family, RTF</td>
</tr>
<tr>
<td>Specify whether to force a new page before printing the table</td>
<td>NEWPAGE on page 628</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify the number of sets of columns to place on a page</td>
<td>PANELS= on page 629</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td>Specify the number of blank characters to place between sets of columns when PANELS= is in effect</td>
<td>PANEL_SPACE= on page 629</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the number of lines that must be available on the page in order to print the body of the table</td>
<td>REQUIRED_SPACE= on page 630</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td>Specify the number of lines to place between the previous output object and the current one</td>
<td>TOP_SPACE= on page 631</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td>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</td>
<td>WRAP on page 632</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td>Specify whether to add a double space after the last line of a single row when the row is wrapped</td>
<td>WRAP_SPACE on page 632</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td><strong>Influence the layout of rows and columns</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the maximum number of blank characters to place between columns</td>
<td>COL_SPACE_MAX= on page 625</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the minimum number of blank characters to place between columns</td>
<td>COL_SPACE_MIN= on page 625</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the name of the column whose value provides formatting information about the space before each row of the template</td>
<td>CONTROL= on page 625</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to double space between the rows of the table</td>
<td>DOUBLE_SPACE on page 626</td>
<td>LISTING</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Specify whether extra space is evenly divided among all columns of the table</td>
<td>EVEN on page 626</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to split a long stacked column across page boundaries</td>
<td>SPLIT_STACK on page 630</td>
<td>LISTING</td>
</tr>
<tr>
<td>Influence the display of the values in header cells and data cells</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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</td>
<td>CLASSLEVELS= on page 625</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td>Specify which format to use if both a column template and a data component specify a format</td>
<td>DATA_FORMAT_OVERRIDED on page 626</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether to justify the format fields within the columns or to justify the values within the columns without regard to the format fields</td>
<td>JUSTIFY on page 627</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to order the columns by their order in the data component</td>
<td>ORDER_DATA on page 628</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify the source of the values for the format width and the decimal width if they are not specified</td>
<td>USE_FORMAT_DEFAULTS on page 632</td>
<td>All</td>
</tr>
<tr>
<td>Use the column name as the column header if neither the column template nor the data component specifies a header</td>
<td>USE_NAME on page 632</td>
<td>All</td>
</tr>
<tr>
<td>Influence the layout of headers and footers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the number of blank lines to place between the last row of data and the first row of output</td>
<td>FOOTER_SPACE= on page 627</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the number of blank lines to place between the last row of headers and the first row of data</td>
<td>HEADER_SPACE= on page 627</td>
<td>LISTING</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>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</td>
<td>OVERLINE on page 628</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to print table footers and any overlining of the table footers</td>
<td>PRINT_FOOTERS on page 629</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to print table headers and any underlining of the table headers</td>
<td>PRINT_HEADERS on page 629</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>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</td>
<td>UNDERLINE on page 631</td>
<td>LISTING</td>
</tr>
</tbody>
</table>

### Influence the non-LISTING output

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify whether to place the output object in a table of contents, if you create a table of contents</td>
<td>CONTENTS on page 625</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify the label to use for the output object in the contents file, the Results window, and the trace record</td>
<td>CONTENTS_LABEL= on page 625</td>
<td>HTML, PDF, PRINTER, PS PDFMARK</td>
</tr>
</tbody>
</table>

### Other table attributes

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control whether BY lines are printed above each BY group</td>
<td>BYLINE= on page 624</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Define the characters to use as the line-drawing characters in the table</td>
<td>FORMCHAR= on page 627</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify a label for the table</td>
<td>LABEL= on page 628</td>
<td>All</td>
</tr>
<tr>
<td>Specify the table that the current template inherits from</td>
<td>PARENT= on page 629</td>
<td>All</td>
</tr>
<tr>
<td>Specify the style element to use for the table and any changes to the attributes</td>
<td>STYLE= on page 630</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>Specify the special data set type of a SAS data set</td>
<td>TYPE= on page 631</td>
<td>OUTPUT</td>
</tr>
</tbody>
</table>
### 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, at SAS invocation, in the OPTIONS statement, or in the 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 BY line 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.

**Example** “Example 1: Creating a Stand-Alone Style” on page 459

### 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.
Restriction 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 8.4 Values in the Control Column

<table>
<thead>
<tr>
<th>Column Control Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A digit from 1-9</td>
<td>The specified number of blank lines precedes the current row.</td>
</tr>
<tr>
<td>A hyphen (-)</td>
<td>A row of underlining precedes the current row.</td>
</tr>
<tr>
<td>&quot;b&quot; or &quot;B&quot;</td>
<td>ODS tries to insert a panel break if the entire table does not fit in the space that is provided.</td>
</tr>
</tbody>
</table>

Default None
Restriction The "b" and "B" column control values are not supported for the PRINTER destination.
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
  use 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.

Examples
  “Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551
  “Example 3: Creating a New Table Template ” on page 563

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.

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.

Tips     Use any character in formatting characters, including hexadecimal characters. If you use hexadecimal characters, then put an x after the closing quotation mark. For example, this option assigns the hexadecimal character 2-D to the first formatting character, the hexadecimal character 7C to the second character, and does not alter the remaining characters: formchar="2D7C"x
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

**Interactions**

JUSTIFY=ON can interfere with decimal alignment.

If the column is numeric, then values are aligned to the right if you specify JUSTIFY=OFF and JUST=C.

All of the destinations except for the LISTING destination justify the values in columns as if JUSTIFY=ON for JUST=R and JUST=L.

**Tips**

If you translate numeric data to character data, you might need to use JUSTIFY= to align the data.

The JUSTIFY attribute is valid only in the LISTING destination.

**See**

"Values in Table Columns and How They Are Justified" on page 548

**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, or the first spanning header in the table.

**Tip**
The LABEL= attribute is valid in all destinations.

**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

**Defaults**

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

**Tip**

The OVERLINE attribute is valid only in the LISTING destination.

**See**

For information about formatting characters, see the discussion of “FORMCHAR= "string" | variable;” on page 627.

UNDERLINE= table attribute on page 631, UNDERLINE= column attribute on page 607, and the OVERLINE= column attribute on page 603.

**Example**

"Example 1: Editing a Table Template That a SAS Procedure Uses" on page 551

**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 line size, display the data in multiple sets of columns so that rows that would otherwise appear on multiple pages appear on the same page.

**Tips**

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).

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

**Tip** The PRINT_FOOTERS attribute is valid in all destinations except the OUTPUT destination.

**See** OVERLINE= table attribute on page 628

**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. For individual columns, use the following column attribute: “PRINT_HEADERS<<ON | OFF | variable>;” on page 605.

**Tip** The PRINT_HEADERS attribute is valid in all destinations except the OUTPUT destination.

**See** “UNDERLINE<<ON | OFF | variable>;” on page 631

**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 448). 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. SAS does 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 "Viewing the Contents of a Style" on page 429

"Finding and Viewing the Default Style for ODS Destinations" on page 429

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**style-attribute-specification** describes the style attribute to change.

Each **style-attribute-specification** has this general form:

\[
\text{style-attribute-name} = \text{style-attribute-value}
\]

See “About Style Attributes” on page 459

---

### Default

**Table**

### Requirement

Specify either a **style-attribute-specification** or a **style-element-name** with the STYLE= option.

### Tips

You can use braces ({ and }) instead of square brackets ([ and ]).

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.

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 a special type of SAS data set.

**Restriction** PROC TEMPLATE and PROC ODSSTABLE do not verify that a SAS data set type that you specify is a valid data set type or the structure of the data set that you create is appropriate for the type that you have assigned.
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.

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

**Tip**

The UNDERLINE attribute is valid only in the LISTING destination.

**See**

For information about formatting characters, see "FORMCHAR=" string | variable;" on page 627.

UNDERLINE= column attribute on page 607 and OVERLINE= column attribute on page 603.

Also see OVERLINE table attribute on page 628.

**Examples**

"Example 1: Editing a Table Template That a SAS Procedure Uses" on page 551

"Example 3: Creating a New Table Template " on page 563

### 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 or PROC ODSTABLE 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 heading if neither the column template nor the data component specifies a header.

**Default**

OFF

**Tips**

Use this attribute when column names are derived from a data set and the columns are generic.

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

<table>
<thead>
<tr>
<th>Interaction</th>
<th>When ODS wraps the rows of a table, it does not place multiple values in any column that contains an ID column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>The WRAP attribute is valid only in LISTING and printer family destinations.</td>
</tr>
<tr>
<td>See</td>
<td>WRAP_SPACE table attribute on page 632 and ID= column attribute on page 601</td>
</tr>
</tbody>
</table>

**WRAP_SPACE**

<table>
<thead>
<tr>
<th>variable &gt;</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>OFF</td>
</tr>
<tr>
<td>Tip</td>
<td>The WRAP_SPACE attribute is valid only in the LISTING, printer family, and RTF destinations.</td>
</tr>
<tr>
<td>See</td>
<td>WRAP= table attribute on page 632</td>
</tr>
</tbody>
</table>

---

### CELLSTYLE AS Statement

For tables, sets the style element of the cells in the table or column according to the values of the variables. For text, sets the style attributes of the list items or paragraphs. Use this statement to set the presentation characteristics (such as foreground color and font face) of individual cells or text.

**Restriction:**
The CELLSTYLE AS statement can be used only within a column template, an ODS list, an ODS textblock, or a table template.

**Syntax**

```
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 list item, paragraph, or table cell.
  - If `expression` resolves to TRUE (a nonzero 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 string, 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 about SAS functions, see [SAS Functions and CALL Routines: Reference](#).

**built-in variable**

is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

- **_COLUMN_**
  is a column number. Column numbering begins with 1.
  
  **Alias**
  _COL_
  
  **Example**
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_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 CELLSYTL 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.
  
  **Example**
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_ROW_**
  is a row number. Row numbering begins with 1.
  
  **Example**
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_STYLE_**
  is a style element name.
  
  **See**
  For a table of style element names, see Chapter 20, "Style Elements," on page 797.
  
  **Example**
  "Example 6: Creating Master Templates" on page 583
**_VAL_**

is the data value of a cell.

**Tip**

Use _VAL_ to represent the value of the current column.

**Example**

“Example 6: Creating Master Templates” on page 583

**comparison-operator**

compares a variable with a value or with another variable.

The following table lists the comparison operators:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td></td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td></td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td></td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td></td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td></td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td></td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>

**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. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**See**

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

**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 about the style attributes that you can set in a table template, see “About Style Attributes” on page 459.
Optional Argument

**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 by using the “DEFINE STYLE Statement” on page 448. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

The following style elements are most likely to be used with the CELLSTYLE AS statement:

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataEmphasisFixed
- DataStrong
- DataStrongFixed
- ListItem
- ListItem2
- Paragraph

The style element provides the basis for displaying the cell. Additional style attributes modify the display.

Default: Data

See Chapter 14, “TEMPLATE Procedure,” on page 425

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

---

### COLUMN Statement

Declares a symbol as a column in the table and specifies the order of the columns.

**Restriction:** The COLUMN statement can be used only within a table template.

**Examples:**

- “Example 3: Creating a New Table Template ” on page 563
- “Example 2: Defining Variables with the COLUMN Statement” on page 270

**Syntax**

```plaintext
COLUMN column(s);
```
Required Argument

**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.

**Defaults**

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.

**See**

“Stacking Values for Two or More Variables” on page 550

---

**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.

**Restriction:**

The COMPUTE AS statement can be used only within a column template.

**Example:**

“Example 2: Defining Variables with the COLUMN Statement” on page 270

**Syntax**

**COMPUTE AS expression;**

**Required Argument**

**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 about SAS functions, see *SAS Functions and CALL Routines: Reference*.

**built-in variable**

is a special type 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_`
  
  **Example**  
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- `_DATANAME_`
  is a data-column name.

- `_LABEL_`
  is a column label.
  
  **Example**  
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- `_ROW_`
  is a row number. Row numbering begins with 1.
  
  **Example**  
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- `_STYLE_`
  is a style-element name.
  
  **Example**  
  "Example 6: Creating Master Templates" on page 583

- `_VAL_`
  is the data value of a cell.
  
  **Tip** Use `_VAL_` to represent the value of the current column.
comparison-operator

compares a variable with a value or another variable.

Table 8.6  Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

Tip

The COMPUTE AS statement can alter values in an output object. None of the templates that SAS provides modifies any values. To determine whether a template was provided by SAS, use the “ODS VERIFY Statement” in SAS Output Delivery System: User’s Guide. 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 whether the COMPUTE AS statement alters values. (See “SOURCE Statement” on page 346.)

See

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

Example

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

DEFINE Statement

Creates a template inside a table template.

Restriction: The DEFINE statement can be used only inside a table template.

See:

“DEFINE COLUMN Statement” on page 530
“DEFINE FOOTER Statement” on page 532
**DEFINE HEADER Statement**

**Example:**

```
Example 2: Defining Variables with the COLUMN Statement" on page 270
```

**Syntax**

```
DEFINE <COLUMN | FOOTER | HEADER> template-name </options>;
statements-and-attributes;
END;
```

**Required Argument**

- `template-name`: specifies the name of the new object.

  **Restriction** `template-name` must be a single-level name.

  **Tip** To reference the template that you are creating from another template, create it outside the table template.

**Optional Arguments**

- `COLUMN | FOOTER | HEADER`: specifies the type of template to create.
  - 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-type` is optional if you specify the COLUMN name before the definition.
  - The same is true for headers and footers.

- `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.

**Restriction:**

- The DYNAMIC statement can be used only in the template of an ODS textblock, ODS list, 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.

**Syntax**

```
DYNAMIC variable-name-1 <=value-1<text-1>;
< variable-name-2 <=value-2<text-2>...>>;
```
Required Argument

`variable-name`

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.

Optional Arguments

`value`

sets the value of the variable.

`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.

END Statement

Ends the table template, header template, column template, or footer template.

Syntax

`END;`

FOOTER Statement

Declares a symbol as a footer in the table and specifies the order of the footers.

Example:  “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

Syntax

`FOOTER footer-specification(s);`

Required Argument

`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 and PROC ODSTABLE treat it as the split character.

See "SPLIT= "character" | variable;" on page 617

**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 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.

---

**HEADER Statement**

Declares a symbol as a header in the table and specifies the order of the headers.

**Example:**

“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

**Syntax**

```
HEADER header-specification(s);
```

**Required Argument**

**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 and PROC ODSTABLE treat it as the split character.

See "SPLIT= "character" | variable;" on page 617
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.

Example

“Example 3: Creating a New Table Template ” on page 563

MVAR Statement

Defines a symbol that references a macro variable. ODS uses 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.

Restriction: When replaying an ODS document with PROC DOCUMENT, values created by the MVAR statement must be re-created in the same session that is replaying the document.

Tip: You can use the MVAR statement in the template of an ODS list, ODS textblock, 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.

See: “Example 3: Creating a New Table Template ” on page 563 and “Example 1: Creating a Stand-Alone Style” on page 459

Syntax

MVAR variable-name-1 <='value-1' '<text-1'>>
   < variable-name-2 <='value-2'> '<text-2'…'>;

Required Argument

variable-name

names a macro variable to reference in the template. ODS uses 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 or PROC ODSTABLE 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.
Optional Arguments

value  
sets the default variable value.

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 converts 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.

Restriction:  
The NMVAR statement can be used only in the template of an ODS list, ODS textblock, 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.

See:  
“Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

Syntax  
NMVAR variable-name-1 <='value–1' '<text-1'>>  
< variable-name-2 <='value-2' '<text-2'...'>;  

Required Argument  
variable-name  
names a macro variable to reference in the template. ODS converts 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.

Optional Arguments  
value  
sets the value of the variable.

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.

Restriction: The NOTES statement can be used only in the template of a table, column, header, 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.

See: “Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

Example: “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

Syntax

NOTES 'text';

Required Argument

*text*

provides information about the table.

TRANSLATE INTO Statement

Translates the specified numeric values to other values.

Restrictions: The TRANSLATE INTO statement can be used only in a column template, an ODS list, an ODS textblock, or a table template.

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.

Syntax

TRANSLATE expression-1 INTO expression-2 <, expression-n INTO expression-m>;

Required Arguments

*expression-1*

is an expression that is evaluated for each list item, paragraph, table, or column cell that contains a numeric variable.

If *expression-1* resolves to TRUE (a nonzero 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 about SAS functions, see SAS Functions and CALL Routines: Reference.

built-in variable

is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

_COLUMN_

is a column number. Column numbering begins with 1.

Alias _COL_

Example "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

_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.

Example "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

_ROW_

is a row number. Row numbering begins with 1.

Example "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

_STYLE_

is a style element name.

Example "Example 6: Creating Master Templates" on page 583

_VAL_

is the data value of a cell.

Tip Use _VAL_ to represent the value of the current column.

Example "Example 6: Creating Master Templates" on page 583
A comparison operator compares a variable with a value or with another variable. The following table lists the comparison operators:

**Table 8.7 Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td></td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ^= or != or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td></td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td></td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td></td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td></td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>

**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.

**See**
You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the [WHERE data set option](SAS Data Set Options: Reference) and the section on WHERE-Expression Processing in [SAS Language Reference: Concepts](SAS Language Reference: Concepts).

**Example**
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

`expression-2` is an expression that specifies the value to use in the list, paragraph, or 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 about SAS functions, see *SAS Functions and CALL Routines: Reference*.

**Built-in variable**
a special type 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_
  
  **Example**  
  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

- **_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
  
  **Example**  
  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

- **_ROW_**
  is a row number. Row numbering begins with 1.
  
  **Example**  
  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

- **_STYLE_**
  is a style element name.
  
  **See**  
  For a table of style element names, see Chapter 20, “Style Elements,” on page 797.
  
  **Example**  
  “Example 6: Creating Master Templates” on page 583

- **_VAL_**
  is the data value of a cell.
  
  **Tip**  
  Use _VAL_ to represent the value of the current column.
  
  **Example**  
  “Example 6: Creating Master Templates” on page 583

**comparison-operator**
comparisons a variable with a value or with another variable. The following table lists the comparison operators:
### Table 8.8 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**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 or 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 formatted field, starting at the left. To right-justify the value, use the `JUSTIFY=ON` attribute.

**See**

"JUSTIFY<=ON | OFF | `variable`;" on page 602 column attribute

You can use any expression that can be used in the `WHERE=` data set option. For information about expressions that you can use in the `WHERE` data set option, see the `WHERE data set option` in *SAS Data Set Options: Reference* and the section on `WHERE-Expression Processing in SAS Language Reference: Concepts`.

**Example**

"Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

---

**Usage: ODSTABLE 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 want 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 Sashelp.Tmplmst and any product-dependent item stores.

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.

3. To view the table templates that SAS provides, double-click the item store that contains a table template, such as Base.

4. From the list view, 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.

The SOURCE statement writes the source code for the specified template to the SAS log.

For example, 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 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 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 takes 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 are 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 8.9 Summary of Column Formatting Attributes

<table>
<thead>
<tr>
<th>Specifications Provided by the Column Template</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothing</td>
<td>Format name, width, and number of decimal places are determined by the data component.</td>
</tr>
<tr>
<td>Format name</td>
<td>Format name and width are determined by the column template; number of decimal places is determined by the data component.</td>
</tr>
<tr>
<td>Format name and width</td>
<td>Format name and width are determined by the column template.</td>
</tr>
<tr>
<td>Format name, width and number of decimal places</td>
<td>All three are determined by the column template.</td>
</tr>
<tr>
<td>Width</td>
<td>No name is specified; width is determined by the column template; number of decimal places is determined by the data component.</td>
</tr>
<tr>
<td>Number of decimal places</td>
<td>No name is specified; width is determined by the data component; number of decimal places is determined by the column template.</td>
</tr>
</tbody>
</table>

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:

Usage: ODSTABLE Procedure

267
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);

Examples: ODSTABLE Procedure

Example 1: Customizing Columns

Features:  COLUMN statement
            DATA= option
            DEFINE statement
            PROC ODSTABLE statement
            Table attributes
            STYLE
            HEADER
            FORMAT
            OPTIONS statement
            TITLE statement

Program

options nodate obs=15;
title "Customizing Columns";
proc odstable data=sashelp.class;
column age sex height weight;
define age;
    style={fontsize=10pt just=l borderrightstyle=dashed background=yellow};
    header='Age of Student';
    format=3.;
end;
define sex;
    style={fontsize=10pt just=l borderrightstyle=dashed};
    header='Gender';
end;
define height;
    style={fontsize=10pt just=l foreground=blue borderrightstyle=dashed};
    header='Height';
end;
run;

Program Description

Specify the SAS system options and titles.

options nodate obs=15;
title "Customizing Columns";

Begin the ODSTABLE procedure and specify the columns. The PROC ODSTABLE statement begins the procedure. The DATA= option specifies the input data set. The COLUMN statement specifies the columns.

proc odstable data=sashelp.class;
column age sex height weight;
define age;
    style={fontsize=10pt just=l borderrightstyle=dashed background=yellow};
    header='Age of Student';
    format=3.;
end;
define sex;
    style={fontsize=10pt just=l borderrightstyle=dashed};
    header='Gender';
end;
define height;
    style={fontsize=10pt just=l foreground=blue borderrightstyle=dashed};
    header='Height';
end;
run;

The following output uses the default HTMLBlue style, with customized columns.
Output 8.1 Customizing Columns

Example 2: Defining Variables with the COLUMN Statement

Features:
- COLUMN statement
- DEFINE statement
- COMPUTE AS statement

Details

The COLUMN statement indicates which data columns are available for use anywhere in the template. A variable must be specified in the COLUMN statement for its value to be available. If you do not want a specific variable and column to appear in the output, you can suppress the variable by using the PRINT=OFF column attribute.
This example calculates a three percent holdback based on the invoice price and manufacturers suggested retail price.

Program

```sas
title "Dealer Profit Assuming Three Percent Holdback";
proc odstable data=sashelp.cars(obs=15);
  column Make Model Msrp Holdback;
  define holdback;
    compute as (msrp * .03) ;
    label="3% Dealer Holdback";
    format=dollar10.2;
  end;
  define msrp;
    print=off;
  end;
run;
```

Program Description

**Specify the title and begin the ODSTABLE procedure.**
```
title "Dealer Profit Assuming Three Percent Holdback";
proc odstable data=sashelp.cars(obs=15);
```

**Specify the data columns to include in the table.** The column statement specifies the columns that are available to be used in the template. Make, Model, and Msrp are in the data set. The Holdback column values will be calculated.
```
column Make Model Msrp Holdback;
```

**Create the Holdback data values.** The define statement creates a column named Holdback. The COMPUTE AS statement specifies the calculation to be used. The LABEL and FORMAT column attributes specify the label and format that the values use.
```
define holdback;
  compute as (msrp * .03) ;
  label="3% Dealer Holdback";
  format=dollar10.2;
end;
```

**Suppress the printing of the Msrp column.** You must include the Msrp variable in the COLUMN statement so that the values are available for calculating the Holdback values. To suppress the printing of the Msrp column, use the DEFINE statement and the PRINT=OFF column attribute.
```
define msrp;
  print=off;
end;
run;
```
### Example 3: Creating and Storing a Customized Table Template

**Features:**
- PROC ODSTABLE statement
- DATA= option
- NAME= option
- STORE= option
- COLUMN statement
- DEFINE statement
- Table attributes
  - STYLE
  - HEADER
  - FORMAT
- OPTIONS statement
- TITLE statement

#### Dealer Profit Assuming Three Percent Holdback

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>3% Dealer Holdback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acura</td>
<td>MDX</td>
<td>$1,108.35</td>
</tr>
<tr>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>$714.60</td>
</tr>
<tr>
<td>Acura</td>
<td>TSX 4dr</td>
<td>$809.70</td>
</tr>
<tr>
<td>Acura</td>
<td>TL 4dr</td>
<td>$995.85</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>$1,312.65</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>$1,383.00</td>
</tr>
<tr>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>$2,692.95</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>$778.20</td>
</tr>
<tr>
<td>Audi</td>
<td>A41.8T convertible 2dr</td>
<td>$1,078.20</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>$955.20</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>$1,002.90</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>$1,034.40</td>
</tr>
<tr>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>$1,099.20</td>
</tr>
<tr>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>$1,189.20</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>$1,274.70</td>
</tr>
</tbody>
</table>

**Output 8.2 Output Showing Selected Column Variables**

---

# Output 8.2 Output Showing Selected Column Variables

---
Details

This example creates a table with customized columns. With the ODSTABLE procedure, you can give the table template a name and store it in the template store of your choice.

Program

```sas
options nodate obs=15;
title;
proc odstable data=sashelp.class name=odstableExample store=Sasuser.MyExampleTemplates;
column age sex height weight;
define age;
  style={fontsize=10pt just=l borderrightstyle=dashed background=yellow};
  header='Age of Student';
  format=3.;
end;
define sex;
  style={fontsize=10pt just=l borderrightstyle=dashed};
  header='Gender';
end;
define height;
  style={fontsize=10pt just=l foreground=blue borderrightstyle=dashed};
  header='Height';
end;
run;
```

Program Description

Specify the SAS system options and titles.

```sas
options nodate obs=15;
title;
```

Specify the name of the new table template and the template store. The DATA= option specifies the input data set. The NAME= option specifies the name of the table template. The STORE= option specifies the template store to put the template in. If the template store specified by the STORE= option does not exist, the STORE= option creates it.

```sas
proc odstable data=sashelp.class name=odstableExample store=Sasuser.MyExampleTemplates;
column age sex height weight;
```

Specify the columns. The COLUMN statement specifies the columns.

```sas
column age sex height weight;
```

Customize the columns. The DEFINE statements modify the specified column. The HEADER table attribute specifies the text for the column heading. The STYLE table argument specifies font size, font color, justification, and line style.

```sas
define age;
  style={fontsize=10pt just=l borderrightstyle=dashed background=yellow};
  header='Age of Student';
  format=3.;
end;
define sex;
```
Example 4: Using PROC TEMPLATE and PROC ODSTABLE

Features:

- PROC ODSTABLE statement:
  - STORE= option
  - NAME= option
- CELLSTYLE-AS statement
- PROC TEMPLATE
  - DEFINE TABLE statement
- ODS PDF statement
- OPTIONS statement
- PROC SQL
- TITLE statement

Details

The following programs create the same output using two different methods. Using the PROC ODSTABLE statement is the same as using the combination of PROC TEMPLATE and DEFINE TABLE statements.

This example also creates a master table template. Master templates are applied globally to all of your tabular output. For more information about master table templates, see "Base.Template.Table" on page 535.

Program: Creating a Table with PROC ODSTABLE

```
ods path reset;
```
ods path show;
ods html close;

options nodate;
ods pdf file="ProcOdstableTable.pdf";
title "Using PROC ODSTABLE";
proc odstable name=Base.Template.Table;
  define header myheader1;
    text "Use the CELLSTYLE-AS Statement to Customize Output"
    style={color=red};
  end;
  define header myheader2;
    text "Use PROC ODSTABLE to Create a Table Template"
    style={color=red};
  end;
  define footer myfooter;
    text "This output is formatted with a master template.";
    style={color=blue};
  end;
  cellstyle _row_ in (13) as {BackgroundColor=Palegreen },
    _row_ in (12) as {BorderBottomStyle=Solid },
    _row_ in (11) as {BackgroundColor=Limegreen },
    mod(_row_,2) as {BackgroundColor=Honeydew},
    _col_ = 1 as {Width=1.2in BorderLeftColor=Black},
    _col_ = 2 as {Width=2in},
    _col_ = 3 as {Width=.6in},
    _col_ = 4 as {Width=.5in},
    _col_ = 5 as {Width=.9in};
run;
proc sql;
  select * from sashelp.class;
run;
quit;
proc odstable name=Base.Template.Table store=mystore;
  cellstyle _row_ in (16) as {BackgroundColor=Palegreen },
    _row_ in (15) as {BorderBottomStyle=Solid },
    _row_ in (14) as {BackgroundColor=Limegreen },
    mod(_row_,2) as {BackgroundColor=Honeydew},
    _col_ = 1 as {Width=1.2in BorderLeftColor=Black},
    _col_ = 2 as {Width=2in},
    _col_ = 3 as {Width=.6in},
    _col_ = 4 as {Width=.5in},
    _col_ = 5 as {Width=.9in};
run;
ods path (prepend) Mystore;
ods path show;
proc sql;
  select * from sashelp.class;
run;
quit;
Program Description

Set ODS path to default settings and display the current ODS path.

```sas
ods path reset;
ods path show;
ods html close;
```

Set the SAS system options, specify the PDF destination, and specify a title.

```sas
options nodate;
ods pdf file="ProcOdstableTable.pdf";
title "Using PROC ODSTABLE";
```

Create the table template `Base.Template.Table`. The PROC ODSTABLE statement creates the master template `Base.Template.Table`. This template is applied to every table created by SAS, unless it is overridden by another template created by PROC TEMPLATE or PROC ODSTABLE, removed with the DELETE statement, or manually removed from the item store. Because there is no STORE= option specified in the PROC ODSTABLE statement, the template is stored in the Sasuser.Templat template store.

```sas
proc odstable name=Base.Template.Table;
```

Create two headers and a footer. The DEFINE statement with “header” template type specified, creates a header. The DEFINE statement with “footer” template type specified, creates a footer. DEFINE statement blocks must end with an END statement.

```sas
define header myheader1;
  text "Use the CELLSTYLE-AS Statement to Customize Output";
  style={color=red};
end;
define header myheader2;
  text "Use PROC ODSTABLE to Create a Table Template";
  style={color=red};
end;
define footer myfooter;
  text "This output is formated with a master template.";
  style={color=blue};
end;
```

Format cells. 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.

```sas
CellStyle_row_in (13) as {BackgroundColor=Palegreen },
_row_in (12) as {BorderStyle=Solid },
_row_in (11) as {BackgroundColor=Limegreen },
```
mod(_row_,2) as {Background=Honeydew},
_col_ = 1 as {Width=1.2in BorderLeftColor=Black},
_col_ = 2 as {Width=2in},
_col_ = 3 as {Width=.6in},
_col_ = 4 as {Width=.5in},
_col_ = 5 as {Width=.9in};
run;

Select columns. The SQL procedure selects all columns from the Sashelp.Class data set.

proc sql;
    select * from sashelp.class;
run;
quit;

Create a second table template named Base.Template.Table. The PROC ODSTABLE statement creates the master template Base.Template.Table. This template is applied to every table created by SAS, unless it is overridden by another template created by PROC TEMPLATE or PROC ODSTABLE, removed with the DELETE statement, or manually removed from the item store. Because the STORE= option is specified in the PROC ODSTABLE statement, the template is placed in the template store Work.Mystore.

proc odstable name=Base.Template.Table store=mystore;

Format cells. 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.

cellstyle _row_ in (16) as {BackgroundColor=Palegreen },
_row_ in (15) as {BorderBottomStyle=Solid },
_row_ in (14) as {BackgroundColor=Limegreen },
mod(_row_,2) as {Background=Honeydew},
_col_ = 1 as {Width=1.2in BorderLeftColor=Black},
_col_ = 2 as {Width=2in},
_col_ = 3 as {Width=.6in},
_col_ = 4 as {Width=.5in},
_col_ = 5 as {Width=.9in};
run;

Add Work.Mystore to the beginning of the ODS search order path. The ODS PATH statement with PREPEND specified adds Work.Mystore to the beginning of the ODS search order path. This enables the template Base.Template.Table from the Work.Mystore item store to be used. The ODS PATH statement displays the modified ODS path.

ods path (prepend) Mystore;
ods path show;

Select columns, close the PDF destination, and open the HTML destination. The SQL procedure selects all columns from the Sashelp.Class data set. The PDF statement closes the PDF destination. The ODS HTML statement opens the HTML statement.

proc sql;
    select * from sashelp.class;
run;
quit;

ods pdf close;
ods html;

**Delete the master templates.** The DELETE statement deletes the master templates. If you do not delete them, they are applied to all of your tabular output until you do delete them.

```plaintext
proc template;
   delete base.template.table;
   delete base.template.table / store=mystore;
run;
```

**Set the ODS path to default settings.**

```plaintext
ods path reset;
```

The following output was created with PROC ODSTABLE. The headers and footers are created by the DEFINE statements.

*Output 8.4  Output Created with PROC ODSTABLE, with Headers and Footers*

---

**Using PROC ODSTABLE**

*Use the CELLSTYLE=AS Statement to Customize Output*

*Use PROC ODSTABLE to Create a Table Template*

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69</td>
<td>112.5</td>
</tr>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>56.5</td>
<td>84</td>
</tr>
<tr>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>65.3</td>
<td>98</td>
</tr>
<tr>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
</tr>
<tr>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>83</td>
</tr>
<tr>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
</tr>
<tr>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>12</td>
<td>59</td>
<td>99.5</td>
</tr>
<tr>
<td><strong>Joyce</strong></td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
</tr>
<tr>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>90</td>
</tr>
<tr>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>56.3</td>
<td>77</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
</tr>
<tr>
<td>Philip</td>
<td>M</td>
<td>16</td>
<td>72</td>
<td>150</td>
</tr>
<tr>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>64.8</td>
<td>128</td>
</tr>
<tr>
<td>Ronald</td>
<td>M</td>
<td>15</td>
<td>67</td>
<td>133</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
</tr>
</tbody>
</table>

*This output is formatted with a master template.*

The following output was created with PROC ODSTABLE. There are no headers and footers for this output, because the DEFINE statement was not used in the second PROC ODSTABLE block.
Program: Creating a Table with PROC TEMPLATE and the DEFINE TABLE Statement

ods path reset;
ods path show;
ods html close;
options nodate;
ods pdf file="ProctemplTable.pdf";
title "Using PROC TEMPLATE and the DEFINE TABLE Statement";
proc template;
  define table Base.Template.Table;
    define header myheader1;
      text "Use the CELLSTYLE-AS Statement to Customize Output";
      style={color=red};
    end;
    define header myheader2;
      text "Use PROC TEMPLATE to Create a Table Template";
      style={color=red};
    end;
define footer myfooter;
Program Description

Set ODS path to default settings and display the current ODS path.

ods path reset;
ods path show;
proc sql;
    select * from sashelp.class;
run;
quit;
ods pdf close;
ods html;
proc template;
    delete base.template.table;
    delete base.template.table / store=mystore;
run;

Set the SAS system options, specify the PDF destination, and specify a title.
options nodate;
ods pdf file="ProctemplTable.pdf";
title "Using PROC TEMPLATE and the DEFINE TABLE Statement";

Create the table template Base.Template.Table. The DEFINE TABLE statement in PROC TEMPLATE creates the master template Base.Template.Table. This template is applied to every table created by SAS, unless it is overridden by another template created by PROC TEMPLATE or PROC ODSTABLE, removed with the DELETE statement, or manually removed from the item store. Because there is no STORE= option specified in the DEFINE TABLE statement, the template is stored in the Sasuser.Templat template store.

proc template;
define table Base.Template.Table;

Create two headers and a footer. The DEFINE statement with “header” template type specified, creates a header. The DEFINE statement with “footer” template type specified, creates a footer. DEFINE statement blocks must end with an END statement.

define header myheader1;
text "Use the CELLSTYLE-AS Statement to Customize Output";
style={color=red};
end;
define header myheader2;
text "Use PROC TEMPLATE to Create a Table Template";
style={color=red};
end;
define footer myfooter;
text "This output is formated with a master template.";
style={color=blue};
end;

Format cells. 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.

cellstyle _row_ in (13) as {BackgroundColor=Palegreen },
_row_ in (12) as {BorderBottomStyle=Solid },
_row_ in (11) as {BackgroundColor=Limegreen },
_mod(_row_,2) as {BackgroundColor=Honeydew},
_col_ = 1 as {Width=1.2in BorderLeftColor=Black},
_col_ = 2 as {Width=2in},
_col_ = 3 as {Width=.8in},
_col_ = 4 as {Width=.5in},
_col_ = 5 as {Width=.9in};
end;
run;

Select columns. The SQL procedure selects all columns from the Sashelp.Class data set.

proc sql;
    select * from sashelp.class;
run;
quit;

Create a second table template named Base.Template.Table. The DEFINE TABLE statement in PROC TEMPLATE creates the master template Base.Template.Table. This template is applied to every table created by SAS, unless
it is overridden by another template created by PROC TEMPLATE or PROC
ODSTABLE, removed with the DELETE statement, or manually removed from the
item store. Because the STORE= option is specified in the DEFINE TABLE
statement, the template is placed in the template store Work.Mystore.

```
proc template;
  define table Base.Template.Table / store=Mystore;
Format cells. 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.

  cellstyle _row_ in (16) as {BackgroundColor=Palegreen },
  _row_ in (15) as {BorderColorStyle=Solid },
  _row_ in (14) as {BackgroundColor=Limegreen },
  mod(_row_,2) as {Background=Honeydew},
  _col_ = 1 as {Width=1.2in BorderLeftColor=Black},
  _col_ = 2 as {Width=2in},
  _col_ = 3 as {Width=.6in},
  _col_ = 4 as {Width=.5in},
  _col_ = 5 as {Width=.9in};
end;
run;
```
Add Work.Mystore to the beginning of the ODS search order path. The ODS
PATH statement with PREPEND specified adds Work.Mystore to the beginning of
the ODS search order path. This enables the template Base.Template.Table from
the Work.Mystore item store to be used. The ODS PATH statement displays the
modified ODS path.

```
ods path (prepend) Mystore;
ods path show;
```
Select columns, close the PDF destination, and open the HTML destination.
The SQL procedure selects all columns from the Sashelp.Class data set. The ODS
PDF statement closes the PDF destination, and the ODS HTML statement opens
the HTML destination.

```
proc sql;
  select * from sashelp.class;
run;
quit;
ods pdf close;
ods html;
```
Delete the templates. The DELETE statements delete the template
Base.Template.Table from the Sasuser.Templat template store and the
Base.Template.Table from the Work.Mystore template store

```
proc template;
  delete base.template.table;
  delete base.template.table / store=mystore;
run;
```
The following output was created with PROC TEMPLATE and the DEFINE TABLE
statement. The headers and footers are created by the DEFINE statements.
Using PROC TEMPLATE and the DEFINE TABLE Statement

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69</td>
<td>112.5</td>
</tr>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>56.5</td>
<td>84</td>
</tr>
<tr>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>65.3</td>
<td>98</td>
</tr>
<tr>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
</tr>
<tr>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>83</td>
</tr>
<tr>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
</tr>
<tr>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>12</td>
<td>59</td>
<td>99.5</td>
</tr>
<tr>
<td>Joyce</td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>56.5</td>
</tr>
<tr>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>90</td>
</tr>
<tr>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>56.3</td>
<td>77</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
</tr>
<tr>
<td>Philip</td>
<td>M</td>
<td>16</td>
<td>72</td>
<td>150</td>
</tr>
<tr>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>64.8</td>
<td>128</td>
</tr>
<tr>
<td>Ronald</td>
<td>M</td>
<td>15</td>
<td>67</td>
<td>133</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>86</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
</tr>
</tbody>
</table>

This output is formatted with a master template.

The following output was created with PROC TEMPLATE and the DEFINE TABLE statement. There are no headers and footers for this output, because the DEFINE statement was not used in the second PROC ODS TABLE block.
After both templates are created, either with PROC TEMPLATE or PROC ODSTABLE, you can view them in the Templates window. The template store Work.Mystore is created by the option STORE=Mystore. The template store Sasuser.Templat is the template store that is created when you do not explicitly create or name your own template store. Sasuser.Templat is created when you specify the PROC ODSTABLE or DEFINE TABLE statement with no STORE= option.
Output 8.8  Templates Window Showing the Table Templates
The ODSTEXT Procedure
Overview: ODSTEXT Procedure

What Does the ODSTEXT Procedure Do?

The ODSTEXT procedure is used to create text block templates. These text block templates create lists and paragraphs for your output. You can use style attributes and formats to customize your content, and WHERE expressions to select your content. With PROC ODSTEXT, you can use the DATA= option to bind your data to the template without using a DATA step.

PROC ODSTEXT can be used with any output destination. However, the procedure is essential for creating content for the ODS destination for PowerPoint and e-books. See “ODS EPUB Statement” in SAS Output Delivery System: User’s Guide and “ODS POWERPOINT Statement” in SAS Output Delivery System: User’s Guide.
for more information about creating output for the ODS destination for PowerPoint and e-book output.

Syntax: ODSTEXT Procedure

PROC ODSTEXT <CONTENTS=text-string><DATA=SAS-data-set>
<NAME=template-name>
    <PAGEBREAK=NO | YES> <PRINT><STORE=template-store>;<CELLSTYLE expression-1 AS <style-element-name>[style-attribute-specification(s)]>
    <, expression-n AS <style-element-name>[style-attribute-specification(s)]>;
DYNAMIC variable-name-1 <=value–1'<text-1'>>
    < variable-name-2 <=value-2'<text-2'...>>;
HEADING<n> "heading-text" / <STYLE=style-override>
MVAR variable-name-1 <=value–1'<text-1'>>
    < variable-name-2 <=value-2'<text-2'...>>;
NMVAR variable-name-1 <=value–1'<text-1'>>
    < variable-name-2 <=value-2'<text-2'...>>;
LIST / <START=integer-value><STYLE=style-override> ;
ITEM <expression> / <FORMAT=format-name>
    <STYLE=style-override>
    <VALUE=integer-value>;
END;
P expression/ <FORMAT=format-name> <STYLE=style-override> ;
TRANSLATE expression-1 INTO expression-2 <, expression-n INTO expression-n>;

PROC ODSTEXT Statement

Creates a text block.

Syntax

PROC ODSTEXT <CONTENTS=text-string><DATA=SAS-data-set>
<NAME=template-name>
    <PAGEBREAK=NO | YES> <PRINT><STORE=template-store>; Optional Arguments

CONTENTS="text-string"
specifies the title for the table of contents. The CONTENTS= option overrides the generated table of contents text.

Tip text-string is the text that can be seen in the PDF table of contents and in the Results Window folder descriptions.
DATA=SAS-data-set
  specifies a SAS data set.
  Tip  No output is produced if you specify the NAME= option without either the 
       DATA= option or the PRINT option. Without the DATA= or PRINT options, 
       ODS creates the template but does not render the output.

NAME=<template-name>
  specifies that the content of the procedure should be saved as a template with 
  the specified name.
  Tip  No output is produced if you specify the NAME= option without either the 
       DATA= option or the PRINT option. Without the DATA= or PRINT options, 
       ODS creates the template but does not render the output.

PAGEBREAK= NO | YES
  specifies whether the procedure should generate a page break.
  NO
      specifies that no page break is generated.
      Alias  OFF
  YES
      specifies that a page break is generated.
      Alias  ON
  Default  NO

PRINT
  specifies that the template is printed as well as stored.
  Restriction  The PRINT option is needed only if the NAME= option is specified 
               without the DATA= option.
  Tip  No output is produced if you specify the NAME= option without either the 
       DATA= option or the PRINT option. Without the DATA= or PRINT options, 
       ODS creates the template but does not render the output.

STORE=template-store
  specifies the template store where the compiled template is placed. If you do not 
  specify a name or template store, ODS stores the template in the first writable 
  template store in the ODS path. By default, this template store is 
  Sasuser.Templat.

  If you do specify a name and template store, but no libref is specified, then the 
  template is stored in the Work directory. For example, the statement proc 
  ods text name=mystore store=mystore; results in a template store named 
  Work.Mystore.
  Restriction  The STORE= option can only be specified if the NAME= option is 
               specified.
CELLSTYLE AS Statement

For tables, sets the style element of the cells in the table or column according to the values of the variables. For text, sets the style attributes of the list items or paragraphs. Use this statement to set the presentation characteristics (such as foreground color and font face) of individual cells or text.

Restriction: The CELLSTYLE AS statement can be used only within a column template, an ODS list, an ODS textblock, or a table template.

Syntax

```plaintext
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 list item, paragraph, or table cell.

If `expression` resolves to TRUE (a nonzero 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:

```plaintext
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 string, 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 about SAS functions, see SAS Functions and CALL Routines: Reference.

`built-in variable`

is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

```plaintext
_COLUMN_
```

is a column number. Column numbering begins with 1.

Alias

```plaintext
_COL_
```
**Example**  
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_**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.

**Example**  
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_**ROW**_  
is a row number. Row numbering begins with 1.

**Example**  
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_**STYLE**_  
is a style element name.

**See**  
For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**Example**  
“Example 6: Creating Master Templates” on page 583

_**VAL**_  
is the data value of a cell.

**Tip**  
Use _**VAL**_ to represent the value of the current column.

**Example**  
“Example 6: Creating Master Templates” on page 583

**comparison-operator**  
compares a variable with a value or with another variable.

The following table lists the comparison operators:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or == or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

**Table 9.1**  
Comparison Operators
Symbol | Mnemonic Equivalent | Definition
--- | --- | ---
> | GT | Greater than
< | LT | Less than
| | | Greater than or equal to
| | | Less than or equal to
| | | Equal to one or more 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. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

See | You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

Example | “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

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 about the style attributes that you can set in a table template, see “About Style Attributes” on page 459.

Optional Argument

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 by using the “DEFINE STYLE Statement” on page 448. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

The following style elements are most likely to be used with the CELLSTYLE AS statement:

- Data
- DataFixed
- DataEmpty
The style element provides the basis for displaying the cell. Additional style attributes modify the display.

**Default**  Data

**See**  Chapter 14, “TEMPLATE Procedure,” on page 425

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

---

**DYNAMIC Statement**

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

**Restriction:**

The DYNAMIC statement can be used only in the template of an ODS textblock, ODS list, 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.

**Syntax**

\[ \text{DYNAMIC } \text{variable-name-1 } \langle= \text{value-1}\langle\text{text-1}\rangle\rangle \\
< \text{variable-name-2 } \langle= \text{value-2}\langle\text{text-2}\rangle\rangle; \]

**Required Argument**

variable-name

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.

**Optional Arguments**

value

sets the value of the variable.

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.
END Statement

Ends item and list blocks.

Syntax

END;

HEADING Statement

Creates H# tags for the ODS TEXT output.

Alias: H

Accessibility note: Adding headings to your output helps make the output accessible to screen reader users and to people with certain visual impairments. For an example, see Create an Accessible Table.

Syntax

HEADING <n> "heading-text" / <STYLE=style-override>;

Required Argument

_heading-text

specifies the heading text. You can use in-line formatting and style overrides to customize your heading text. This option helps to make reports accessible for people with a wide range of abilities.


For information about style overrides, see “STYLE= <style-element-name >[style-attribute-name=style-attribute-value<… style-attribute-name=style-attribute-value> ]” on page 297.

Optional Arguments

_n

specifies the heading level.

Range 1–6

Example The following statement creates a second-level heading:

h2 "My Heading 2";
STYLE=<style-element-name>[style-attribute-name=style-attribute-value]<...style-attribute-name=style-attribute-value>]
specifies the style element to use for the heading text. For example, the following statement specifies that the text color for the third heading is red:

H3 'Third Heading' / style=[color=red];

Note: You can use braces ({ and }) instead of square brackets ([ and ]).

style-element-name
is the name of a style element that is part of a style template that is registered with the Output Delivery System. SAS provides some style templates. You can create your own style templates with PROC TEMPLATE.


For a list of style elements, see Chapter 20, “Style Elements,” on page 797.

style-attribute-name
specifies the attribute to change.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

style-attribute-value
specifies a value for the attribute. Each attribute has a different set of valid values.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

ITEM Statement
Specifies an item to be added to the item list. An ITEM statement that does not specify an expression begins an item block.

Restriction: ITEM statements that do not specify an expression must end with an END statement.

Syntax
ITEM <expression> / <FORMAT=format-name> <STYLE=style-override> <VALUE=integer-value>;

Optional Arguments
expression
is an expression that specifies the content of an item.

expression has this form:

expression-1 < expression-n>
expression is an arithmetic or logical sequence of operators and operands. An operator is a symbol that requests a logical operation, a string, or an arithmetic calculation.

An operand is one of the following:

constant is a fixed value, such as the name of a column, a text string, or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

function specifies a SAS function. For information about SAS functions, see SAS Functions and CALL Routines: Reference.

Restriction ITEM statements that do not specify an expression must end with an END statement.

FORMAT=\texttt{format-name} specifies a default format for the value in each table cell or list item. You can use any SAS or user-defined format.

STYLE=\texttt{<style-element-name >[style-attribute-name=style-attribute-value<… style-attribute-name=style-attribute-value>]} specifies the style element to use for the specified item. For example, the following statement specifies that the text color for the item is red:

\begin{verbatim}
item 'first item' / style=[color=red];
\end{verbatim}

Note: You can use braces ({ and }) instead of square brackets ([ and ]).

\texttt{style-element-name} is the name of a style element that is part of a style template that is registered with the Output Delivery System. SAS provides some style templates. You can create your own style templates with PROC TEMPLATE.

See Chapter 14, "TEMPLATE Procedure," on page 425 for information about PROC TEMPLATE and the default style templates.

For a list of style elements, see Chapter 20, “Style Elements,” on page 797.

\texttt{style-attribute-name} specifies the attribute to change.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

\texttt{style-attribute-value} specifies a value for the attribute. Each attribute has a different set of valid values.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

\texttt{VALUE=integer-value} starts a numbered list from the specified integer-value. If another VALUE= option is specified in subsequent statements, numbering begins again at that point.
Restriction
When `VALUE=` is specified, you must also specify a numbering list type with the `LISTSTYLETYPE=` style attribute. For example, `LISTSTYLETYPE= "decimal_leading_zero"` or `LISTSTYLETYPE="decimal"`.

Tip
You can change the bullet type of a list item by specifying the `"LISTSTYLETYPE=bullet-type"` style attribute with the `STYLE=` option.

Example
“Example 2: Creating Nested Lists” on page 216

LIST Statement
Creates a list.

Restriction:
LIST statements begin a LIST statement block, and must end with an END statement.

Example:
“Example 2: Comparing Lists Created with PROC ODSLIST and PROC ODSTEXT” on page 313

Syntax
```
LIST / <START=integer-value><STYLE=style-override> ;
```

Optional Arguments

**START=integer-value**
starts a numbered list from the specified `integer-value`. If another `START=` option is specified in subsequent statements, the numbering begins again at that point.

Restrictions
When `START=` is specified, you must also specify a numbering list type with the `LISTSTYLETYPE=` style attribute. For example, `LISTSTYLETYPE= "decimal_leading_zero"` or `LISTSTYLETYPE="decimal"` are numbering list types.

The `START=` option is valid for the PDF, POWERPOINT, HTML, and RTF destinations.

Tip
You can change the bullet type of a list item by specifying the `"LISTSTYLETYPE=bullet-type"` style attribute with the `STYLE=` option.

Examples
The following example creates a list that begins numbering items with the number 7. For HTML5 output, if you want to start a numbered list from a specific number with the `START=` option, then specify `LISTSTYLETYPE=` in the LIST statement, as this example does.
```
proc odstext;
  p 'This is a numbered list';
  list / style={liststyletype="decimal_leading_zero"} start=7;
    item "Seven";
    item "Eight";
  end;
```
The following example creates a list that assigns a specific number to each item.

```
proc odstext;
  p 'This is a numbered list';
  list / style={liststyle=`decimal_leading_zero`};
    item "Two" / value=2;
    item "Four" / value=4;
    item "Six" / value=6;
  end;
run;
```

**STYLE=<style-element-name >[style-attribute-name=style-attribute-value]<...**

Specifies the style element to use for the items in the list.

For example, the following statement specifies that the text color for item is red:

```
item 'first item' / style=[color=red];
```

**Note:** You can use braces {{ and }} instead of square brackets [[ and ]].

---

**MVAR Statement**

Defines a symbol that references a macro variable. ODS uses 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.

**Restriction:** When replaying an ODS document with PROC DOCUMENT, values created by the MVAR statement must be re-created in the same session that is replaying the document.
You can use the MVAR statement in the template of an ODS list, ODS textblock, 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.

See: “Example 3: Creating a New Table Template ” on page 563 and “Example 1: Creating a Stand-Alone Style” on page 459

Syntax

MVAR variable-name-1 <=value-1' <text-1>'>  
< variable-name-2 <=value-2'> <text-2'...'>>;

Required Argument

variable-name  
names a macro variable to reference in the template. ODS uses 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 or PROC ODSTABLE 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.

Optional Arguments

value  
sets the default variable value.

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 converts 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.

Restriction:  
The NMVAR statement can be used only in the template of an ODS list, ODS textblock, 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.

See: “Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

Syntax

NMVAR variable-name-1 <=value-1' <text-1'>>  
< variable-name-2 <=value-2'> <text-2'...'>>;
Required Argument

`variable-name`

names a macro variable to reference in the template. ODS converts 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.

Optional Arguments

`value`

sets the value of the variable.

`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.

P Statement

Specifies a paragraph. Multiple P statements are allowed within an item block.

Restriction:  In the ODSLIST Procedure, the P statement can be specified only within an ITEM block. In the ODSTEXT procedure, the P statement can be used at the top level of PROC ODSTEXT as well as within list items.

Syntax

```
P expression / <FORMAT=format-name> <STYLE=style-override> ;
```

Required Argument

`expression`

is an expression that specifies the content of an item or paragraph. `expression` has this form:

```
expression-1 < expression-n>
```

`expression`

is an arithmetic or logical sequence of operators and operands. An operator is a symbol that requests 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, text string, or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

For example, the following code creates a paragraph and an item list with PROC ODSTEXT:
proc odslist data=sashelp.class;
  item;
    p "My name is " || name;
  list;
    item;
      p "My age is " || put(age, 2.);
      p "My weight is " || put(weight, 3.);
    end;
  end;
run;

The following code creates an item list with PROC ODSLIST:

proc odslist data=sashelp.class;
  item;
    p "My name is " || name;
  list;
    item;
      p "My age is " || put(age, 2.);
      p "My weight is " || put(weight, 3.);
    end;
  end;
run;

*function*

specifies a SAS function. For information about SAS functions, see *SAS Functions and CALL Routines: Reference*.

**Restriction**

ITEM statements that do not specify an expression must end with an END statement.

Optional Arguments

**FORMAT=**format-name

specifies a default format for the value in each paragraph. You can use any SAS or user-defined format.

**STYLE=**<style-element-name >[style-attribute-name=style-attribute-value<... style-attribute-name=style-attribute-value>]

specifies the style element to use for the items in the list or paragraph.

**Note:** You can use braces ({ and }) instead of square brackets ([ and ]).

**style-element-name**

is the name of a style element that is part of a style template that is registered with the Output Delivery System. SAS provides some style templates. You can create your own style templates with PROC TEMPLATE.

See Chapter 14, "TEMPLATE Procedure," on page 425 for information about PROC TEMPLATE and the default style templates.

For a list of style elements, see Chapter 20, “Style Elements,” on page 797.

**style-attribute-name**

specifies the attribute to change.
See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

**style-attribute-value**
specifies a value for the attribute. Each attribute has a different set of valid values.

See For information about style attributes and their values, see Chapter 21, “Style Attributes,” on page 827.

**Examples**
For example, the following statement specifies that the text color of the paragraph is red:
```
p 'text block' / style=[color=red];
```

For example, the following statement specifies that the text color for the item is red:
```
item 'first item' / style=[color=red];
```

## TRANSLATE INTO Statement

Translates the specified numeric values to other values.

**Restrictions:**
The TRANSLATE INTO statement can be used only in a column template, an ODS list, an ODS textblock, or a table template.
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.

### Syntax

```
TRANSLATE expression-1 INTO expression-2 <, expression-n INTO expression-m>;
```

**Required Arguments**

**expression-1**
is an expression that is evaluated for each list item, paragraph, table, or column cell that contains a numeric variable.

If `expression-1` resolves to TRUE (a nonzero 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 about SAS functions, see [SAS Functions and CALL Routines: Reference](#).

**built-in variable**

is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

- **_COLUMN_**
  - is a column number. Column numbering begins with 1.
  - **Alias**  
    - _COL_
  - **Example**  
    - "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_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.
  - **Example**  
    - "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_ROW_**
  - is a row number. Row numbering begins with 1.
  - **Example**  
    - "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_STYLE_**
  - is a style element name.
  - **Example**  
    - "Example 6: Creating Master Templates" on page 583

- **_VAL_**
  - is the data value of a cell.
  - **Tip**  
    - Use _VAL_ to represent the value of the current column.
  - **Example**  
    - "Example 6: Creating Master Templates" on page 583

**comparison-operator**

compares a variable with a value or with another variable. The following table lists the comparison operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal to</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
</tr>
<tr>
<td><code>=</code></td>
<td>equal to</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>not equal to</td>
</tr>
</tbody>
</table>
Table 9.2  Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>

**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.

**See**
You can use any expression that can be used in the `WHERE=` data set option. For information about expressions that you can use in the `WHERE` data set option, see the `WHERE` data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

`expression-2`

is an expression that specifies the value to use in the list, paragraph, or 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 about SAS functions, see SAS Functions and CALL Routines: Reference.
**Built-in variable**

A special type 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_
  
  **Example**  
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_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
  
  **Example**  
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_ROW_**
  is a row number. Row numbering begins with 1.
  
  **Example**  
  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

- **_STYLE_**
  is a style element name.
  
  **See**  
  For a table of style element names, see Chapter 20, "Style Elements," on page 797.
  
  **Example**  
  "Example 6: Creating Master Templates" on page 583

- **_VAL_**
  is the data value of a cell.
  
  **Tip**  
  Use _VAL_ to represent the value of the current column.
  
  **Example**  
  "Example 6: Creating Master Templates" on page 583

**Comparison-operator**

Compares a variable with a value or with another variable. The following table lists the comparison operators:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ~= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>Symbol</td>
<td>Mnemonic Equivalent</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

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 or 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 formatted field, starting at the left. To right-justify the value, use the JUSTIFY=ON attribute.

See: “JUSTIFY<=ON | OFF | variable>;” on page 602 column attribute

You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

Example: “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

Usage: ODSTEXT Procedure

Using the ODSTEXT Procedure

PROC ODSTEXT uses many of the same statements as PROC TEMPLATE to create and customize your text. The CELLSTYLE AS, DYNAMIC, MVAR, NMVAR, and TRANSLATE INTO statements are all valid statements within PROC ODSTEXT that can be used to customize your text.
Changing the Bullet Type for Lists

By default, lists created by the ODS TEXT procedure are displayed as an unordered list with a disc as the bullet type. In markup family output, you can change the bullet type for lists by using the following two steps:

1. Specify the `LISTSTYLETYPE=` style attribute with the desired bullet type in a style override. The following shows the general syntax for specifying the `LISTSTYLETYPE=` attribute::

   ```math
   STYLE={ LISTSTYLETYPE="bullet-type" <more-style-attributes> }
   ```

2. Specify the style override in an ITEM or LIST statement. The following shows the general syntax for specifying a style override in the LIST or ITEM statements:

   ```math
   LIST / STYLE={ LISTSTYLETYPE="bullet-type" <more-style-attributes> }
   ITEM / STYLE={ LISTSTYLETYPE="bullet-type" <more-style-attributes> }
   ```

Note:
For HTML5 output, if you want to start a numbered list from a specific number with the `START=` option, then specify `LISTSTYLETYPE=` in the LIST statement.

**IMPORTANT**
The `LISTSTYLETYPE=` style attribute is valid in the Markup family, Excel, and PowerPoint destinations. The valid bullet-type values differ for each destination. For more information about valid bullet types, see the `LISTSTYLETYPE=` Style Attribute on page 877.

When you specify `LISTSTYLETYPE=` in the LIST statement, the bullets for all of the following items are changed. If you specify `LISTSTYLETYPE=` in an ITEM statement, only the bullet for that ITEM statement is changed.

The following example creates a numbered list. Because the style override is specified in the LIST statement, all of the items will have the same numbering scheme.

```math
ods html5 file='entireList_html5.htm';
title;
proc odstext;
p 'Follow these steps to change the bullet type for lists:'
   / style={fontsize=12pt};
list / style={liststyletype="decimal" fontsize=12pt};
item "Decide what bullet type to display.";
item "Add a '/' after the LIST statement (if not already present).";
item "Specify the LISTSTYLETYPE= attribute with the STYLE= option.";
end;
run;
ods _all_ close;
```
You can change the bullet type for each individual item by specifying a style override in each ITEM statement.

```
ods html5 file='individualItems_html5.htm';
   title;
   proc odstext;
   p 'You can change the bullet type for each individual item.'
      / style={fontsize=12pt};
   list / style={fontsize=12pt};
   item "List items can be a square" / style={liststyletype="square"};
   item "Or a circle" / style={liststyletype="circle"};
   item "Or a disc" / style={liststyletype="disc"};
   item "Or items can have no bullet" / style={liststyletype="none"};
   end;
   run;
   ods _all_ close;
```

Here are the browser-supported bullet types. However, many browsers support other common types and the support is browser dependent. If an unsupported bullet type is specified, the bullet’s display is browser dependent.

**Table 9.4**  
**Valid Bullet Types**

<table>
<thead>
<tr>
<th>Bullet Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered Lists</td>
<td></td>
</tr>
<tr>
<td>Bullet Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>decimal</td>
<td>The list items are ordered with numbers (default).</td>
</tr>
<tr>
<td>upper_latin</td>
<td>The list items are ordered with uppercase letters.</td>
</tr>
<tr>
<td>lower_latin</td>
<td>The list items are ordered with lowercase letters.</td>
</tr>
<tr>
<td>upper_roman</td>
<td>The list items are ordered with uppercase roman numerals.</td>
</tr>
<tr>
<td>lower_roman</td>
<td>The list items are ordered with lowercase roman numerals.</td>
</tr>
</tbody>
</table>

Unordered Lists

<table>
<thead>
<tr>
<th>Bullet Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disc</td>
<td>The list items are marked with filled circles.</td>
</tr>
<tr>
<td>circle</td>
<td>The list items are marked with circles.</td>
</tr>
<tr>
<td>square</td>
<td>The list items are marked with squares.</td>
</tr>
<tr>
<td>none</td>
<td>The list items are not marked.</td>
</tr>
</tbody>
</table>

Examples: ODSTEXT Procedure

Example 1: Creating Content for the ODS Destination for PowerPoint

Features:

- P statement
  - STYLE= option
- PROC ODSTEXT statement
  - P statement
- FOOTNOTE statement
- ODS HTML CLOSE statement
- ODS POWERPOINT statement
- OPTIONS statement
- TITLE statement
Details

The following example creates text output to add to a Microsoft PowerPoint slide. You can use the STYLE= option in the P statement to customize your text.

Program

```sas
ods html close;
options nodate;
title 'Using PROC ODSTEXT';
footnote 'The ODS Destination for PowerPoint';
ods powerpoint file="layoutTwocontent.ppt" layout=twocontent;

proc odstext;
p 'You can use the ODSTEXT procedure to add paragraphs and lists to your output.';
p 'You can also format your text.' / style=[color=red fontsize=25pt];
p 'This slide shows output created by PROC GMAP.' / style=[color=purple fontsize=30pt];
run;

proc gmap map=maps.us data=maps.us all;
id state;
choro state/statistic=frequency discrete;
run;
quit;
ods _all_ close;
```

Program Description

**Close the HTML destination, specify titles, and specify footnotes.** The ODS HTML destination is open by default in the SAS Windowing environment. If you are not creating HTML output, close the HTML destination to conserve system resources.

```sas
ods html close;
options nodate;
title 'Using PROC ODSTEXT';
footnote 'The ODS Destination for PowerPoint';
```

**Create a file for the ODS destination for PowerPoint.** The ODS POWERPOINT statement creates output formatted for the ODS destination for PowerPoint.

```sas
ods powerpoint file="layoutTwocontent.ppt" layout=twocontent;
```

**Create content for the ODS destination for PowerPoint.**

```sas
proc odstext;
p 'You can use the ODSTEXT procedure to add paragraphs and lists to your output.';
p 'You can also format your text.' / style=[color=red fontsize=25pt];
p 'This slide shows output created by PROC GMAP.'
```
Create PROC GMAP output and close open destinations.

```
proc gmap map=maps.us data=maps.us all;
  id state;
  choro state/statistic=frequency discrete;
run;
quit;

ods _all_ close;
```

Adding Text to Your Output

**Output 9.3** Adding Text to Your Output

Example 2: Comparing Lists Created with PROC ODSLIST and PROC ODSTEXT

Features:

**ODSLIST Procedure**
- CELLSTYLE AS statement
- END statement
- ITEM statement
- LIST statement
- P statement:
  - PROC ODSLIST statement
  - TRANSLATE INTO statement

**ODSTEXT Procedure**
- CELLSTYLE AS statement
- END statement
- ITEM statement
- LIST statement
- P statement
Details

The following examples create similar output using PROC ODSLIST and PROC ODSTEXT. Both procedures can be used to create lists and paragraphs. However, there are differences in how the syntax of the two procedures can be specified. The following table shows how the P, ITEM, and LIST statements are specified in each procedure.

<table>
<thead>
<tr>
<th>PROC ODSTEXT</th>
<th>PROC ODSLIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>The P statement can be specified only within an ITEM block.</td>
<td>The P statement can be used at the top level of PROC ODSTEXT as well as within list items.</td>
</tr>
<tr>
<td>The ITEM statement must be used within a LIST block.</td>
<td>The ITEM statement does not have to be specified within a LIST block.</td>
</tr>
<tr>
<td>The LIST statement does not have to be specified within an item block.</td>
<td>The LIST statement must be used within an item block.</td>
</tr>
</tbody>
</table>

There are also slight differences in the appearance of the output created by the two procedures. Examine the output created by the two programs in this example. Notice that the spacing and bullet points are slightly different.

Program: Using PROC ODSLIST

```plaintext
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";

proc odslist data=sashelp.class;
  item;
    p 'List of Students' Names, Ages, and Weights' / style=systemtitle;
    p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};
  list;
    cellstyle age<=13 as datastrong{background=green},
      age>=14 as {background=lightpurple};
    item 'Student name is ' || name;
    item 'Age: ' || put(age, 2.);
  item;
    translate age<=13 into 'Weight: N/A';
    p 'Weight: ' || put(weight, 3.);
  end;
end;
end;
```
Program Description

Specify the SAS system options and title.

```sas
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";
```

Begin the ODSLIST procedure, begin an ITEM block, and specify explanatory text. The PROC ODSLIST statement specifies the input data set and begins the procedure. The ITEM statement with no options specified begins an ITEM block. The P statements specify the explanatory text. In the ODSLIST Procedure, the P statement must be specified within an ITEM block.

```sas
proc odslist data=sashelp.class;
  item;
    p 'List of Students'' Names, Ages, and Weights' / style=systemtitle;
    p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};
  end;
end;
run;
```

Begin the list and conditionally set the style attributes. The LIST statement with no options specified begins a list block. The LIST statement must be used within an item block. The CELLSTYLE AS statement sets the style attributes of the list items based on the value of the variable Age. If the value of the variable Age is less than or equal to 13, the background color is green. If the value of the variable Age is greater than or equal to 14, the background color is light purple.

```sas
list;
  cellstyle age<=13 as datastrong{background=green},
       age>=14 as {background=lightpurple};
end;
```

Specify the items. The ITEM statements specify the content of each item.

```sas
item 'Student name is ' || name;
item 'Age: ' || put(age, 2.);
```

Conditionally transform the value for the AGE variable. The TRANSLATE INTO statement translates the value of Age based on the conditions specified. For values of the variable AGE that are equal to or less than thirteen, "N/A" is written to the output.

```sas
item;
  translate age<=13 into 'Weight: N/A';
  p 'Weight: ' || put(weight, 3.);
end;
```

End the LIST and ITEM blocks. The LIST block began with the LIST statement. An END statement ends the LIST block. Both ITEM blocks began with an ITEM statement with no expression specified. An END statement ends each ITEM block.

```sas
end;
end;
run;
```
options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";

proc odstext data=sashelp.class;
p 'List of Students' Names, Ages, and Weights' / style=systemtitle;
p 'Weights of children younger than fourteen are hidden for legal reasons' / style={fontstyle=italic};
list;
cellstyle age<=13 as datastrong{background=green},
age>=14 as {background=lightpurple};
item 'Student name is ' || name;
item 'Age: ' || put(age, 2.);
item;
translate age<=13 into 'Weight: N/A';
p 'Weight: ' || put(weight, 3.);
end;
end;
run;

Program Description

Specify the SAS system options and title.
  options nodate nonumber obs=7;
title "Students' Names, Ages, and Weights";

Begin the ODSTEXT procedure and specify explanatory text for the list. The PROC ODSLIST statement specifies the input data set and begins the procedure. The P statements specify the text that precedes the list. In the ODSTEXT procedure, the P statement does not have to be specified within an ITEM block.
proc odstext data=sashelp.class;
  p 'List of Students'' Names, Ages, and Weights' / style=systemtitle;
  p 'Weights of children younger than fourteen are hidden for legal reasons' /
     style={fontstyle=italic};

Begin the list and conditionally set the style attributes. The LIST statement with
no options specified begins a list block. In the ODESTEXT procedure, the LIST
statement does not have to be specified within an item block. The CELLSTYLE AS
statement sets the style attributes of the list items based on the value of the variable
Age. If the value of the variable Age is less than or equal to 13, the background
color is green. If the value of the variable Age is greater than or equal to 14, the
background color is light purple.

list;
  cellstyle age<=13 as datastrong{background=green},
     age>=14 as {background=lightpurple};

Specify the items. The ITEM statements specify the content of each item.

item 'Student name is ' || name;
item 'Age: ' || put(age, 2.);

Conditionally transform the value for the AGE variable. The TRANSLATE INTO
statement translates the value of Age based on the conditions specified. For values
of the variable AGE that are equal to or less than thirteen, "N/A" is written to the
output. The TRANSLATE INTO statement must be specified within the block of the
variable you want to be translated. Otherwise, the variable is translated and the
results apply to all variables.

item;
  translate age<=13 into 'Weight: N/A';
  p 'Weight: ' || put(weight, 3.);
end;

End the LIST block. The LIST block began with the LIST statement. An END
statement ends the LIST block.

end;
run;

Example 2: Comparing Lists Created with PROC ODSLIST and PROC ODSTEXT
# Output 9.5

Output Created with the PROC ODS\TEXT

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is hidden</strong></td>
</tr>
<tr>
<td>Age: 14</td>
</tr>
<tr>
<td>Weight: 113</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is Alice</strong></td>
</tr>
<tr>
<td>Age: 13</td>
</tr>
<tr>
<td>Weight: 105</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is Barbara</strong></td>
</tr>
<tr>
<td>Age: 13</td>
</tr>
<tr>
<td>Weight: 103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is Carol</strong></td>
</tr>
<tr>
<td>Age: 14</td>
</tr>
<tr>
<td>Weight: 103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is Henry</strong></td>
</tr>
<tr>
<td>Age: 14</td>
</tr>
<tr>
<td>Weight: 103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is James</strong></td>
</tr>
<tr>
<td>Age: 13</td>
</tr>
<tr>
<td>Weight: 103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Students’ Names, Ages, and Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights of children younger than fourteen are hidden for legal reasons</td>
</tr>
<tr>
<td><strong>Student name is Jane</strong></td>
</tr>
<tr>
<td>Age: 13</td>
</tr>
<tr>
<td>Weight: 104</td>
</tr>
</tbody>
</table>
PART 7

The TEMPLATE Procedure

Chapter 10
  Overview .................................................. 321

Chapter 11
  TEMPLATE Procedure: Managing Template Stores ........... 333

Chapter 12
  TEMPLATE Procedure: Creating Crosstabulation Table Templates ... 359

Chapter 13
  TEMPLATE Procedure: Creating ODS Graphics ................. 419

Chapter 14
  TEMPLATE Procedure: Creating a Style Template ............ 425

Chapter 15
  TEMPLATE Procedure: Creating Tabular Templates ........... 515

Chapter 16
  Tabular Attributes ....................................... 591

Chapter 17
  TEMPLATE Procedure: Creating Markup Language Tagsets ...... 633
Overview

Introduction to 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

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.

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.

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 item stores, Sasuser.Registry and Sashelp.Registry.

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 15, “TEMPLATE Procedure,” on page 515.

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.

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.

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 10.1  Compatibility of ODS Templates between SAS Versions

<table>
<thead>
<tr>
<th>ODS Template</th>
<th>Backward Compatible</th>
<th>Forward Compatible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Crosstabs</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Style</td>
<td>No</td>
<td>Yes *</td>
</tr>
<tr>
<td>ODS Template</td>
<td>Backward Compatible</td>
<td>Forward Compatible</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Tagset</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ODS Graphics</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

* Styles that use inheritance might not be compatible forwards or backward. See “Inheritance Compatibility across Versions” on page 444 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.

## Syntax

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
Using the TEMPLATE Procedure

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 the HTML output

For the code that creates the following default and customized output, see “Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551.
Modify a Style

When you are working with styles, you are more likely to modify a style that SAS supplies than to write a completely new style. The following output uses the Styles.HTMLBlue template that SAS provides, but includes changes made to the style in order to customize the output's appearance. For the code that creates this output, see “Example 3: Modifying the Default Style with the CLASS Statement” on page 474.

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.

Figure 10.1  HTML Output (Viewed with Microsoft Internet Explorer)

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`.

To see the customized CHTML tagset, view the source from your web browser by selecting View ⇒ Source from your browser’s toolbar.
Create a Template-Based Graph

STATGRAPH templates are used to create output called ODS Graphics. For complete information, see SAS Graph Template Language: User’s Guide.

The following code creates the STATGRAPH template MyGraphs.Regplot, which creates the following graph.

```sas
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;
end;
```
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.

*Figure 10.3 Graph Created with a STATGRAPH Template*

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. For the SAS code that creates this output, see “Example 2: Creating a Crosstabulation Table Template with a Customized Legend” on page 402.

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:
PROC TEMPLATE Statements by Category

This table lists and describes the categories and statements used in the TEMPLATE procedure.

**Table 10.2 Categories and PROC TEMPLATE Statements**

<table>
<thead>
<tr>
<th>Task</th>
<th>Statements Category</th>
<th>Statements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate</td>
<td>Template store</td>
<td>DELETE</td>
<td>Deletes the specified template</td>
</tr>
<tr>
<td>Task</td>
<td>Statements Category</td>
<td>Statements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>and manage ODS templates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINK</td>
<td></td>
<td>LINK</td>
<td>Creates a link to an existing template</td>
</tr>
<tr>
<td>LIST</td>
<td></td>
<td>LIST</td>
<td>Lists items in one or more template stores</td>
</tr>
<tr>
<td>PATH</td>
<td></td>
<td>PATH</td>
<td>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</td>
</tr>
<tr>
<td>SOURCE</td>
<td></td>
<td>SOURCE</td>
<td>Writes the source code for the specified template</td>
</tr>
<tr>
<td>TEST</td>
<td></td>
<td>TEST</td>
<td>Tests the most recently created template by binding it to the specified data set</td>
</tr>
<tr>
<td>Create or modify ODS style</td>
<td>Style</td>
<td>DEFINE STYLE</td>
<td>Creates a style for any destination that supports the STYLE= option</td>
</tr>
<tr>
<td>Create and modify ODS tabular</td>
<td>Tabular</td>
<td>EDIT</td>
<td>Edits an existing template</td>
</tr>
<tr>
<td>Define column</td>
<td></td>
<td>DEFINE COLUMN</td>
<td>Creates a template for a column</td>
</tr>
<tr>
<td>Define footer</td>
<td></td>
<td>DEFINE FOOTER</td>
<td>Creates a template for a table footer</td>
</tr>
<tr>
<td>Define header</td>
<td></td>
<td>DEFINE HEADER</td>
<td>Creates a template for a header</td>
</tr>
<tr>
<td>Define table</td>
<td></td>
<td>DEFINE TABLE</td>
<td>Creates a template for a table</td>
</tr>
<tr>
<td>Create or modify markup</td>
<td>Markup language tagsets</td>
<td>DEFINE TAGSET</td>
<td>Creates a template for a tagset</td>
</tr>
<tr>
<td>Task</td>
<td>Statements Category</td>
<td>Statements</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>language tagsets</td>
<td></td>
<td>DEFINE CROSSTAB S</td>
<td>Creates a template for a PROC FREQ crosstabulation table</td>
</tr>
<tr>
<td>Create or modify a crosstabulation table template</td>
<td>Tabular</td>
<td>DEFINE STATGRAPH</td>
<td>Creates a template for a graph</td>
</tr>
</tbody>
</table>

### Where to Go from Here

Creating statistical graphics with ODS:
For reference information about the Graph Template Language, see *SAS Graph Template Language: Reference*.

Creating statistical graphics with ODS:
For usage information about PROC TEMPLATE and the Graph Template Language, see *SAS 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 11, “TEMPLATE Procedure,” on page 333.

Modifying an existing style or creating your own style:
For reference information about the style template statements in PROC TEMPLATE, see Chapter 14, “TEMPLATE Procedure,” on page 425.

Creating and modifying ODS tabular output:
For reference information about the tabular template statements in PROC TEMPLATE, see Chapter 15, “TEMPLATE Procedure,” on page 515.

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 17, “TEMPLATE Procedure,” on page 633.
Overview: TEMPLATE Procedure: Managing Template Stores

What Is a Template Store?

A template store is an item store that 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-
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

Concepts: TEMPLATE Procedure: Managing Template Stores

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.
In the Results window, select the Results folder. Right-click to open the Templates window.

To view the definitions or templates that SAS supplies, click the plus sign that is next to the Sashelp.Tmplmst item store.

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**

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.

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 346.

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.
Figure 11.1 Templates That SAS Supplies

Syntax: TEMPLATE Procedure: Managing Template Stores

**PROC TEMPLATE;**

**DELETE item-path< / STORE=libref.Template-store >;**

**LINK item-path-1 TO item-path-2 < /options>;**

**LIST <starting-path>/options>;**

**PATH location(s);**

**SOURCE item-path < /options><STORE=libref.Template-store>;**

**TEST DATA=data-set< / STORE=libref.Template-store>;**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROC TEMPLATE</strong></td>
<td>Begin a PROC TEMPLATE template</td>
<td>Ex. 1, Ex. 2, Ex. 3</td>
</tr>
<tr>
<td><strong>DELETE</strong></td>
<td>Delete the specified item</td>
<td></td>
</tr>
<tr>
<td><strong>LINK</strong></td>
<td>Create a link to an existing item</td>
<td></td>
</tr>
<tr>
<td><strong>LIST</strong></td>
<td>List items in one or more template stores</td>
<td>Ex. 1, Ex. 2</td>
</tr>
</tbody>
</table>
### Statement Table

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PATH</strong></td>
<td>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</td>
<td>Ex. 1, Ex. 2, Ex. 3</td>
</tr>
<tr>
<td><strong>SOURCE</strong></td>
<td>Write the source code for the specified item to the SAS log</td>
<td>Ex. 3</td>
</tr>
<tr>
<td><strong>TEST</strong></td>
<td>Test the most recently created item by binding it to the specified data set</td>
<td></td>
</tr>
</tbody>
</table>

### PROC TEMPLATE Statement

Begins a PROC TEMPLATE template.

#### Syntax

```latex
PROC TEMPLATE;
  DELETE item-path < / STORE=libref.template-store >;
  LINK item-path-1 TO item-path-2 <options>;
  LIST <starting-path><options>;
  PATH location(s);
  SOURCE item-path <options><STORE=libref.template-store>;
  TEST DATA=data-set < / STORE=libref.template-store>;
```

### DELETE Statement

Deletes the specified item.

Examples:

- “Example 2: Comparing the EDIT Statement to the DEFINE TABLE Statement” on page 557
- “Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551

#### Syntax

```latex
DELETE item-path < / STORE=template-store >;
```

#### Required Argument

- **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.

Optional Argument

**STORE= template-store**  
specifies the template store where the template that you want to delete is stored.

### LINK Statement

Creates a link to an existing item.

**Tip:** 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= Statement” on page 454. However, using a link is more efficient than using inheritance, because linking does not actually create a new item.

**Syntax**

```
LINK item-path-1 TO item-path-2 <options>;
```

**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.

**Optional Arguments**

- **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 is 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.

**Example:**
“Example 1: Listing Templates in a Template Store” on page 351

---

**Syntax**

```
LIST <starting-path>/<options>;
```

**Optional Argument**

**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.

**Options**

**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 on which 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 338).
**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 | STORE | (statistic-1 <, ... statistic-n>)**

specifies the information to include in the list of items.

**ALL**

includes all available information.

**STORE**

specifies the name of the template store that contains the items.

**(statistic-1 <, ... statistic-n>)**

includes the specified information. statistic is one or more of the following:

**CREATED**

is the date on which 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 338).

**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  For information about setting the current template path, see the "PATH Statement" on page 344.

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
is a special type 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.

Aliases  NAME | _NAME_

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;

STORE
is the name of the template store that contains the item.

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:
```sas
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:
```sas
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:
```sas
source / where=(size > 70000);
run;
```

CREATED
is the date on which 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:
```sas
source / where=(datepart(created) = today());
```
The DATEPART function extracts the date from a SAS datetime value.

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:
```sas
source / where=(_cdate_ = '16JUL2004'd);
run;
```

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:

```sql
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:

```sql
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:

```sql
source / where=(_mdatetime_ = '01may04:9:30:00'dt);
run;
```

**MODIFIED**

is the date on which 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:

```sql
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:

```sql
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>
<thead>
<tr>
<th>Comparison Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>

---

**LIST Statement**

343
### Table 11.1 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td></td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ^= or ^= or &lt;&gt;</td>
<td>NE</td>
<td></td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td></td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td></td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td></td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td></td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**See** For information about expressions that you can use in the WHERE data set option, see the WHERE data set option and the section on WHERE-Expression Processing.

**Example** “Example 2: Using a WHERE Expression to Select Items in a Template Store” on page 353

---

### 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.

**Examples:**

- “Example 1: Listing Templates in a Template Store” on page 351
- “Example 3: Viewing the Source of a Template” on page 355

**Syntax**

```plaintext
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 in which 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 \langle(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). SAS stores all the items that it provides in
Sashelp.Tmplmst.

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.

See “RSASUSER System Option” in SAS System Options: Reference for
more information about how to open the Sasuser library for Read
access or Read-Write access.

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).
Optional Argument

**(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.

**Example:**

“Example 3: Viewing the Source of a Template” on page 355

**Syntax**

```
SOURCE item-path <options>;
```

**Required Argument**

**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 the “NOTES Statement” on page 541 and the discussion of the “LINK Statement” on page 338. You can also specify notes as quoted strings in the DYNAMIC, MVAR, NMVAR, REPLACE, and STYLE statements.)
Optional Arguments

**EXPAND**

prints the source of all parents of a template.

**Example**
The following PROC TEMPLATE block prints all of the parents of the template `base.contents.variables`:

```sas
proc template;
    source base.contents.variables / expand;
run;
```

**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 DATA Step Statements: Reference* for information about the FILENAME statement.

**NOFOLLOW**

specifies that the program does not resolve links in the PARENT= statement, which specifies the item that the current item inherits from. For information about the PARENT= statement, see "PARENT= Statement" on page 454 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. If there is no STORE= option, when you run the code, then the item that it creates goes 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:

```sas
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:

```sas
(subsetting-variable <comparison-operator where-expression-n>)
```
**subsetting-variable**

A special type 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.

**Aliases**

| NAME | _NAME_
| TEMPLATE | _TEMPLATE_

**Example**

This SOURCE statement displays the code for all items that contain the word "Default" in the name of the current template:

```javascript
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:

```javascript
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:

```javascript
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:

```sas
source / where=(size > 70000);
run;
```

**CREATED**

is the date on which 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:

```sas
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:

```sas
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:

```sas
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:

```sas
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:

```sas
source / where=(_mdate_ = '16JUL2004'd);
run;
```
**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:

```sas
source / where=(_mdatetime_ = '01may03:9:30:00'dt);
run;
```

**MODIFIED**

_is the date on which 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:

```sas
source / where=(datepart(modified) = today());
```

>The DATEPART function extracts the date from a SAS datetime value.

**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:

```sas
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>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~</td>
<td>or ^= or &lt;&gt;</td>
<td>NE</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>
TEST Statement

Tests the most recently created item by binding it to the specified data set.

Syntax

```
TEST DATA=data-set < STORE=libref.template-store>
```

Required Argument

**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.

Optional Argument

**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.

Examples: TEMPLATE Procedure: Managing Template Stores

Example 1: Listing Templates in a Template Store

**Features:**
- PATH statement
- LIST statement:
  - starting-path option
  - SORT= option

**Details**

This example lists the items for the Base.Univariate directory in the item store Sashelp.Tmplmst.
Program

    proc template;
    path sashelp.tmplmst;
    list base.univariate / sort=path descending;
    run;

Program Description

**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;
Example 2: Using a WHERE Expression to Select Items in a Template Store

Features:
- PATH statement
- LIST statement:
  - `WHERE` expression option
Details
This example uses a WHERE expression to select, for listing, fitted distribution table templates in the Base.Univariate directory.

Program

```plaintext
proc template;
  path sashelp.tmplmst;
  list base.univariate / sort=path descending
    where=(lowcase(path) ? 'fit');
run;
```

Program Description

**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.

```plaintext
proc template;
  path sashelp.tmplmst;
```

**List, in descending order, the items with the word “fit” in their pathname.** 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 pathname. 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.

```plaintext
list base.univariate / sort=path descending
  where=(lowcase(path) ? 'fit');
run;
```
Example 3: Viewing the Source of a Template

Features:
- PATH statement
- SOURCE statement

Details
This example displays the source code for the Style_popup tagset that SAS provides.

Program
```sas
proc template;
  path sashelp.tmplmst;
  source Tagsets.Style_popup;
run;
```
Program Description

**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 Style_popup that SAS provides. The source code is written to the SAS log.

```
source Tagsets.Style_popup;
run;
```
Partial Source Code of the Template Tagset.Style_Popup That Is Written to the SAS Log

```sas
proc template;
   NOTE: Writing HTML Body file: sashtml.htm
path sashelp.tmplmst;
source Tagsets.Style_popup;
define tagset Tagsets.Style_popup;
   notes "This is HTML pop up styles."
   define event style_class;
      put "." HTMLCLASS " {font-family: " HTMLCLASS "}" NL;
      put "." HTMLCLASS NL;
      trigger stylesheetclass;
      put ")" NL;
      trigger linkclass /if exists( linkcolor);
      trigger vlinkclass /if exists( visitedlinkcolor);
      trigger alinkclass /if exists( activelinkcolor);
   end;
   define event doc_head;
      start:
         put "<head>" NL;
         put VALUE NL;
      finish:
      trigger popup_script;
      put "</head>" NL NL;
   end;
   define event stylesheet_link;
      set $stylesheet_idx "0";
      break /if ^exists( url);
      put "<style type="text/css">" NL "<!--" NL;
      trigger alignstyle;
      put "-->" NL "</style>" NL;
      set $urlList url;
   end;

   ... more log output ...

   %nrstr(" if %( element.parentElement &&
    element.parentElement.parentElement &&
    element.parentElement.parentElement.className %) " NL;
   put "{" NL;
   put " element.style.backgroundColor = '#ff6666';" NL;
   put " element.style.color = 'white';" NL;
   put " element.title = element.parentElement.parentElement.className;" NL;
   put "}" NL;
   put %nrstr(" if %( element.parentElement && element.parentElement.className
   $) " NL;
   put "{" NL;
   put " element.style.backgroundColor = '#ff6666';" NL;
   put " element.style.color = 'white';" NL;
   put " element.title = element.parentElement.className;" NL;
   put "}" NL;
   put " if ( element.className )" NL;
   put "{" NL;
   put " element.style.backgroundColor = '#ff6666';" NL;
   put " element.style.color = 'white';" NL;
   put " element.title = element.className;" NL;
   put "}" NL;
```

Example 3: Viewing the Source of a Template
TEMPLATE Procedure: Creating Crosstabulation Table Templates

Overview: TEMPLATE Procedure: Creating Crosstabulation Table Templates . 359
Using the TEMPLATE Procedure to Create a Customized Crosstabulation Table 360
What Can You Do with a Crosstabulation Template? ................................. 360

Concepts: TEMPLATE Procedure: Creating Crosstabulation Table Templates . 363
What Makes the Crosstabulation Table Unique? ...................................... 363
Comparison between Table Templates and Crosstabulation Table Templates . 363

Syntax: TEMPLATE Procedure: Creating Crosstabulation Table Templates . 364
DEFINE CELLVALUE Statement ......................................................... 365
DEFINE CROSSTABS Statement ...................................................... 368
DEFINE FOOTER Statement .......................................................... 374
DEFINE HEADER Statement .......................................................... 374
CELLSTYLE AS Statement .............................................................. 377
CELLVALUE Statement ................................................................. 380
DYNAMIC Statement ..................................................................... 380
END Statement .............................................................................. 382
FOOTER Statement ....................................................................... 382
HEADER Statement ......................................................................... 383
NOTES Statement .......................................................................... 383
TEXT Statement .............................................................................. 384

Usage: TEMPLATE Procedure: Creating Crosstabulation Table Templates . 387
Working with the CrossTabFreqs Crosstabulation Table Template .......... 387
Crosstabulation Table Regions and Corresponding Attributes .......... 388

Examples: TEMPLATE Procedure: Creating Crosstabulation Table Templates . 390
Example 1: Creating a Customized Crosstabulation Table
Template with No Legend ................................................................. 390
Example 2: Creating a Crosstabulation Table Template with a
Customized Legend .................................................................. 402
Example 3: Adding Custom Formats to Cellvalues .......................... 413
Overview: TEMPLATE Procedure: Creating Crosstabulation Table Templates

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 12.1 PROC TEMPLATE Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a crosstabulation table template</td>
<td>DEFINE CROSSTABS</td>
</tr>
<tr>
<td>Define a value that appears in the crosstabulation cells</td>
<td>DEFINE CELLVALUE</td>
</tr>
<tr>
<td>Specify the order in which the cellvalues are stacked in the cells</td>
<td>CELLVALUE</td>
</tr>
<tr>
<td>Create a template for a footer</td>
<td>DEFINE FOOTER</td>
</tr>
<tr>
<td>Create a template for a header</td>
<td>DEFINE HEADER</td>
</tr>
<tr>
<td>End a crosstabulation table template</td>
<td>END</td>
</tr>
</tbody>
</table>

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:

**Figure 12.1**  Crosstabulation Table Created with Default Crosstabulation Table Template

The following display shows PROC FREQ output that has been created with a modified CrossTabFreqs template:
Figure 12.2  Crosstabulation Table Created with Modified Crosstabulation Table Template

Here are some of the customizations that were made to the preceding crosstabulation table:

- 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 cell values are now red when the deviation exceeds abs (2.0).
- The TotalPercent cell value has been moved from the middle of the other cell values to the bottom of the cell values.
What Makes the Crosstabulation Table Unique?

Crosstabulation tables produced by PROC FREQ are different from other tables that SAS produces. Most other tables consist 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 that is called a legend. The legend contains the labels for the cell values. 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.

Syntax: TEMPLATE Procedure: Creating Crosstabulation Table Templates

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>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELLSTYLE AS</td>
<td>Set the style element of the cells in the column according to the values of the variables</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>CELLVALUE</td>
<td>Specify the order in which the cellvalues are stacked in the cells</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>DEFINE CELLVALUE</td>
<td>Define a value that appears in the crosstabulation cells</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>DEFINE CROSSTABS</td>
<td>Create a crosstabulation table template</td>
<td>Ex. 1, Ex. 2, Ex. 3</td>
</tr>
<tr>
<td>DEFINE FOOTER</td>
<td>Create a template for a footer</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>DEFINE HEADER</td>
<td>Create a template for a header</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>DYNAMIC</td>
<td>Define a symbol that references a value that the data component supplies from the procedure or DATA step</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>END</td>
<td>End a template</td>
<td>Ex. 1, Ex. 2, Ex. 3</td>
</tr>
<tr>
<td>FOOTER</td>
<td>Declare a symbol as a footer in the table and specify the order of the footers</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>HEADER</td>
<td>Declare a symbol as a header in the table and specify the order of the headers</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>NOTES</td>
<td>Provide information about a template</td>
<td>Ex. 1, Ex. 2</td>
</tr>
<tr>
<td>TEXT</td>
<td>Specify the text of the header or the footer</td>
<td>Ex. 1, Ex. 2</td>
</tr>
</tbody>
</table>

**DEFINE CELLVALUE Statement**

Defines a value that appears in the crosstabulation cells.

**Note:**
The DEFINE CELLVALUE statement begins a DEFINE CELLVALUE block. The following statements are commonly used within the block: "CELLSTYLE AS Statement" on page 377, “DYNAMIC Statement” on page 380, “NOTES Statement” on page 383, and “END Statement” on page 382.

**Example:**
“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

**Syntax**

```
DEFINE CELLVALUE <cellvalue>;
    <cellvalue-attribute-1> < cellvalue-attribute-n>;
    CELLSYLE expression-1 AS <style-element-name><style-attribute-specification(s)>;
```
Optional Argument

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 and ATTRIBUTE-NAME=ON.
Table 12.2  DEFINE CELLVALUE Attribute Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify which format to use for the cellvalue if both a crosstabulation template and a data component specify a format</td>
<td>“DATA_FORMAT_OVERRIDE=ON</td>
</tr>
<tr>
<td>Specify the format for the cellvalue</td>
<td>“FORMAT=format_name &lt;format-width&lt;decimal-width&gt;&gt;;” (p. 367)</td>
</tr>
<tr>
<td>Override the width specified by the FORMAT= attribute</td>
<td>“FORMAT_WIDTH=positive-integer” (p. 368)</td>
</tr>
<tr>
<td>Override the number of decimals specified by the FORMAT= attribute</td>
<td>“FORMAT_NDEC=positive-integer” (p. 368)</td>
</tr>
<tr>
<td>Specify the text in the legend</td>
<td>“HEADER='text'” (p. 368)</td>
</tr>
<tr>
<td>For the Output destination, specify the label for the data set column corresponding to the cellvalue</td>
<td>“LABEL='text ’” (p. 368)</td>
</tr>
<tr>
<td>Specify whether a cellvalue appears in the crosstabulation table</td>
<td>“PRINT= ON</td>
</tr>
</tbody>
</table>

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 in the TABLES statement in PROC FREQ, then the data component specifies 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, or the length of the variable for character variables.

Range The minimum cell width is 8, and the maximum width is 25.
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.

**FORMAT_WIDTH=**positive-integer
overides the width specified by the FORMAT= attribute statement.

**FORMAT_NDEC=**positive-integer
overides the number of decimals specified by the FORMAT= attribute statement.

**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.

---

**DEFINE CROSSTABS Statement**

Creates a crosstabulation table template.

**Note:**
The DEFINE CROSSTABS statement begins a crosstabs table template. The following statements are typically used within a DEFINE CROSSTABS block: “**DEFINE CELLVALUE Statement**” on page 365, “**DEFINE HEADER Statement**” on page 374, “**DEFINE FOOTER Statement**” on page 374, “**DYNAMIC Statement**” on page 380, “**FOOTER Statement**” on page 382, “**HEADER Statement**” on page 383, and “**NOTES Statement**” on page 383.

**Example:**
“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

---

**Syntax**

**DEFINE CROSSTABS** table-path ;

< 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;

Summary of Optional Arguments

CELL_STYLE=<style-element-name>[style-attribute-specification(s)] | style-name
Specify a style element and any changes to its attributes to use for the cellvalues in the non-summary rows and columns

COL_TOTAL_STYLE=<style-element-name>[style-attribute-specification(s)]
Specify the style element and any changes to its attributes to use for the cellvalues in the last row in the table

COL_VAR_STYLE=<style-element-name>[style-attribute-specification(s)]
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

COLS_HEADER=header-name
Specify the name of the header to use over the column variable value columns in the table

GRAND_TOTAL_STYLE=<style-element-name>[style-attribute-specification(s)]
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

LABEL="text"
Specify a label for the table

LEGEND_STYLE=<style-element-name>[style-attribute-specification(s)]
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

ROW_TOTAL_STYLE=<style-element-name>[style-attribute-specification(s)]
Specify the style element and any changes to its attributes to use for the cellvalues in the cells that contain row totals

ROW_VAR_STYLE=<style-element-name>[style-attribute-specification(s)]
Specify the style element and any changes to its attributes to use for the row variable values in the leftmost column of the table

ROWS_HEADER=header-name
Specify the name of the header to use over the row variable values (leftmost) column in the table

STORE= template-store
Specify the template store in which to store the crosstabulation template.

STYLE=<style-element-name>[style-attribute-specification(s)]
Specify a style element and any changes to its attributes to use for the table
Required Argument

**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 about template stores, see “Understanding Item Stores and Template Stores” in SAS Output Delivery System: User’s Guide. PROC TEMPLATE writes the template to the first writable template store in the current path.

**Requirement**
Crosstabulation table templates must be named CrossTabFreqs.

Optional Argument

**STORE= template-store**
specifies the template store in which to store the crosstabulation template. If the template store does not exist, it is created.

**Restrictions**
The STORE= option does not become part of the template.

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>The STORE= option must be proceeded by the forward slash (/) symbol.</td>
</tr>
</tbody>
</table>

DEFINE CROSSTABS Attributes

This section lists all of the attributes that you can use in a crosstabulation template.

**Table 12.3  Crosstabulation Attributes**

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a style element and any changes to its attributes to use for the cellvalues in the non-summary rows and columns</td>
<td>CELL_STYLE= (p. 371)</td>
</tr>
<tr>
<td>Specify the name of the header to use over the column variable value columns in the table</td>
<td>COLS_HEADER= (p. 371)</td>
</tr>
<tr>
<td>Specify the style element and any changes to its attributes to use for the cellvalues in the last row in the table</td>
<td>COL_TOTAL_STYLE= (p. 371)</td>
</tr>
<tr>
<td>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</td>
<td>COL_VAR_STYLE= (p. 372)</td>
</tr>
<tr>
<td>Specify the style element and any changes to its attributes to use for the</td>
<td>GRAND_TOTAL_STYLE= (p. 372)</td>
</tr>
<tr>
<td>Task</td>
<td>Statement</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>cellvalues in the rightmost column of the</td>
<td>LABEL= (p. 372)</td>
</tr>
<tr>
<td>last row in the table</td>
<td></td>
</tr>
<tr>
<td>Specify a label for the table</td>
<td>LEGEND_STYLE= (p. 372)</td>
</tr>
<tr>
<td>Specify the style element and any changes</td>
<td>ROWS_HEADER= (p. 372)</td>
</tr>
<tr>
<td>to its attributes to use for the legend</td>
<td>ROW_TOTAL_STYLE= (p. 372)</td>
</tr>
<tr>
<td>table that appears near the upper left</td>
<td>ROW_VAR_STYLE= (p. 373)</td>
</tr>
<tr>
<td>corner of the table</td>
<td></td>
</tr>
<tr>
<td>Specify the name of the header to use</td>
<td></td>
</tr>
<tr>
<td>over the row variable values (leftmost)</td>
<td></td>
</tr>
<tr>
<td>column in the table</td>
<td></td>
</tr>
<tr>
<td>Specify the style element and any changes</td>
<td></td>
</tr>
<tr>
<td>to its attributes to use for the cellvalues</td>
<td></td>
</tr>
<tr>
<td>in the cells that contain row totals</td>
<td></td>
</tr>
<tr>
<td>Specify the style element and any changes</td>
<td></td>
</tr>
<tr>
<td>to its attributes to use for the row</td>
<td></td>
</tr>
<tr>
<td>variable values in the leftmost</td>
<td></td>
</tr>
<tr>
<td>column of the table</td>
<td></td>
</tr>
<tr>
<td>Specify a style element and any changes</td>
<td></td>
</tr>
<tr>
<td>to its attributes to use for the table</td>
<td>STYLE= (p. 373)</td>
</tr>
</tbody>
</table>

**CELL_STYLE=**<style-element-name><[style-attribute-specification(s)]> | style-name  
-----------------------------------|--------------
specifies the style element and any changes to its attributes that 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 "Crosstabulation Table Regions and Corresponding Attributes" on page 388 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 "Crosstabulation Table Regions and Corresponding Attributes" on page 388 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.
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

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

LABEL="text"
specifies a label for the table.

Default: "Frequency Counts and Percentages"

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

ROWS_HEADER=header-name
specifies the name of the header to use over the row variable values (leftmost)
column in the table.

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 “Crosstabulation Table Regions and Corresponding Attributes” on page 388 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 “Crosstabulation Table Regions and Corresponding Attributes” on page 388 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. For more information, see “DEFINE STYLE Statement” on page 448. 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 “Viewing the Contents of a Style” on page 429

“Finding and Viewing the Default Style for ODS Destinations” on page 429

“Crosstabulation Table Regions and Corresponding Attributes” on page 388 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:

See For information about the style attributes, see Chapter 21, “Style Attributes,” on page 827.

“Crosstabulation Table Regions and Corresponding Attributes” on page 388 to see an illustration of the crosstabulation table regions and the DEFINE CROSSTABS attributes that affect each region.

Default Table
DEFINE FOOTER Statement

Creates a footer template.

See: 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 374.

“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

Note: The DEFINE FOOTER statement begins a footer template block. The following statements are commonly used within a DEFINE FOOTER block: “DYNAMIC Statement” on page 380, “NOTES Statement” on page 383, and “TEXT Statement” on page 384.

Syntax

DEFINE FOOTER symbol;
    <attribute-1>< attribute-n>;
    DYNAMIC variable-1<text-1> < variable-n<text-n>>;
    NOTES 'text';
    TEXT header-specification </ expression>;
END;

DEFINE HEADER Statement

Creates a header template.

Note: The DEFINE HEADER statement begins a header template block. The following statements are commonly used within a DEFINE HEADER block: “DYNAMIC Statement” on page 380, “NOTES Statement” on page 383, and “TEXT Statement” on page 384.

See: “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

Syntax

DEFINE HEADER symbol;
    <attribute-1>< attribute-n>;
    DYNAMIC variable-1<text-1> < variable-n<text-n>>;
    NOTES 'text';
    TEXT header-specification </ expression>;
END:

Summary of Optional Arguments

- **CINDENT='character'**
  Specify alignment for headers and footers that wrap.

- **SPACE=positive-integer**
  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.

- **STYLE=<[style-element-specification(s)]>**
  Specify the style element and any changes to its attributes to use for the header or footer.

Required Argument

- **symbol**
  specifies a name to be referenced by the HEADER statement.

**DEFINE HEADER and DEFINE FOOTER Attribute Statements**

This section lists the attributes that you can use in a header or footer template.

_Table 12.4  DEFINE HEADER and DEFINE FOOTER Attribute Statements_

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify alignment for headers and footers that wrap</td>
<td>“CINDENT='character'” (p. 375)</td>
</tr>
<tr>
<td>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</td>
<td>“SPACE=positive-integer” (p. 375)</td>
</tr>
<tr>
<td>Specify the style element and any changes to its attributes to use for the header or footer</td>
<td>“STYLE=&lt;[style-element-specification(s)]&gt;” (p. 376)</td>
</tr>
</tbody>
</table>

**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 open parenthesis:

```plaintext
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.
The SPACE= attribute is valid only in the LISTING destination.

"Example 1: Creating a Customized Crosstabulation Table Template with No Legend" on page 390

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. 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 448. 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 more information, see “Viewing the Contents of a Style” on page 429.

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
**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.

**Restriction:**
The CELLSTYLE AS statement can be used only with the DEFINE CELLVALUE statement.

**See:**
“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

**Syntax**

```
CELLSTYLE expression-1 AS <style-element-name><[style-attribute-specification(s)]>
<, expression-n AS <style-element-name><[style-attribute-specification(s)]>>;
```
Required Argument

**expression**

is an expression that is evaluated for each cell. If **expression** resolves to TRUE (a nonzero 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**

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:

```plaintext
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:

```plaintext
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 12.5  Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or &lt;= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>
Symbol | Mnemonic Equivalent | Definition
---|---|---
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.

Optional Arguments

**style-attribute-specification**

describes a style attribute to set.

Each *style-attribute-specification* has this general form:

```
style-attribute-name = style-attribute-value
```

**Default**

If you do not specify any style attributes to modify, ODS uses the unmodified *style-element-name*.

**Note**

Neither *style-attribute-specification* nor *style-element-name* is required. However, you must use at least one of them.

**See**

For information about the style attributes that you can set in a column template, see Chapter 21, “Style Attributes,” on page 827.

**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. For more information, see “DEFINE STYLE Statement” on page 448. 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
Note Neither style-attribute-specification nor style-element-name is required. However, you must use at least one of them.

See “Viewing the Contents of a Style” on page 429

“Finding and Viewing the Default Style for ODS Destinations” on page 429

### CELLCVALUE Statement

Specifies the order in which the cellvalues are stacked in the cells.

**Interaction:** If a cellvalue symbol that was specified by the DEFINE CELLCVALUE statement is not present in the list, it does not appear in the crosstabulation table.

**See:** “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

### Syntax

```plaintext
CELLVALUE cellvalue(s);
```

**Required Argument**

**cellvalue(s)** specifies one of the nine possible cellvalues created by the DEFINE CELLCVALUE statement. cellvalues are ordered from the top to the bottom.

See “DEFINE CELLCVALUE Statement” on page 365

### DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

**Restriction:** The DYNAMIC statement can be used only with the DEFINE CELLCVALUE, DEFINE HEADER, and DEFINE FOOTER statements.

**Tip:** A dynamic variable that is defined in a template is available to that template and all the templates that it contains.

**Example:** “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

### Syntax

```plaintext
DYNAMIC dynamic-variable(s);
```
Required Argument

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.

Example “Example 1: Creating a Customized Crosstabulation Table
Template with No Legend” on page 390

STRATAVARIABLE NAMES

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.

Requirement The DYNAMIC statement must specify the
STRATAVARIABLELABELS dynamic variables before the
dynamic variables can be used in an expression.

Tip The value is undefined if the table has only one stratum.
**STRATAVARIABLELABELS**

is a string that identifies the current stratum by the label of the stratum variables.

\[ \text{var-1=value-1<var-n=value-n}> \]

\[ \text{var-1–var-n} \]

specifies the stratum variables.

\[ \text{value-1–value-n} \]

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.

**Examples**

“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

“Example 2: Creating a Crosstabulation Table Template with a Customized Legend” on page 402

---

**END Statement**

Ends the crosstabulation template or a DEFINE CELLVALUE, DEFINE HEADER, or DEFINE FOOTER code block.

**Restriction:** The END statement must be used with the DEFINE CELLVALUE, DEFINE HEADER, DEFINE FOOTER, and DEFINE CROSSTABS statements.

**See:**

“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

---

**Syntax**

\[ \text{END;} \]

---

**FOOTER Statement**

Declares a symbol as a footer in the table and specifies the order of the footers.

**Restriction:** The FOOTER statement can be used only within a crosstabulation table template.

**Syntax**

\[ \text{FOOTER footer-specification(s);} \]
Required Argument

**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**

“DEFINE FOOTER Statement” on page 374

---

**HEADER Statement**

Declares a symbol as a header in the table and specifies the order of the headers.

Restriction: The HEADER statement can be used only within a crosstabulation table template.

**Syntax**

```
HEADER header-specification(s);
```

Required Argument

**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.

Restriction: The NOTES statement can be used only with the DEFINE CROSSTABS, DEFINE CELLVALUE, and DEFINE HEADER statements.

**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.

**Syntax**

```
NOTES 'text';
```
Required Argument

'text'
  provides information about the template.

TEXT Statement

Specifies the text of the header or the footer.

Restriction: The TEXT statement can be used only with the DEFINE HEADER or DEFINE FOOTER statements.

Example: “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

Syntax

TEXT header-specification(s) <options>;

Required Argument

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.

Optional Argument

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 (“ ”).

 Rowe 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.

\[ \text{var-1=value-1<var-n=value-n>} \]

\[ \text{var-1–var-n} \]

specifies the stratum variables.

\[ \text{value-1–value-n} \]

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:

\[ \text{var-1=value-1<var-n=value-n?>} \]

\[ \text{var-1–var-n} \]

specifies the stratum variables.

\[ \text{value-1–value-n} \]

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 12.6  Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ~= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

Usage: TEMPLATE Procedure: Creating Crosstabulation Table Templates

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 that 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” in SAS Output Delivery System: User’s Guide.

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.
Most regions use DEFINE CROSSTABS style attributes to specify a style. The following table shows the style attribute that affects each table region. For complete documentation on DEFINE CROSSTABS attributes, see “DEFINE CROSSTABS Attributes” on page 370. Headers and footers use the STYLE= attribute that is valid for the DEFINE HEADER and DEFINE FOOTER statements. For information about the STYLE= attribute, see “DEFINE HEADER and DEFINE FOOTER Attribute Statements” on page 375.

Table 12.7  Table Region and Corresponding Style Attribute

<table>
<thead>
<tr>
<th>Item</th>
<th>Crosstabulation Table Region</th>
<th>Style Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Legend</td>
<td>LEGEND_STYLE=</td>
</tr>
<tr>
<td>2</td>
<td>Row variable name</td>
<td>ROWS_HEADER=</td>
</tr>
<tr>
<td>3</td>
<td>Row variable value</td>
<td>ROW_VAR_STYLE=</td>
</tr>
</tbody>
</table>
### Examples: TEMPLATE Procedure: Creating Crosstabulation Table Templates

#### Example 1: Creating a Customized Crosstabulation Table Template with No Legend

**Features:**
- `crosstabs-attributes` statements
- `CELLVALUE` statement
- `DEFINE CELLVALUE` statement
  - `CELLSTYLE AS` 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
Example 1: Creating a Customized Crosstabulation Table Template with No Legend

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

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
;
ods path (prepend) work.templat(update);
ods noproctitle;
proc template;
   define style white;
      parent=styles.htmlblue;
      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;
   define crosstabs Base.Freq.CrossTabFreqs;
      notes "Crosstabulation table";
      style=table {backgroundcolor=#BFCFFF};
      cell_style=data {backgroundcolor=#FFFFFF};
      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};
Example 1: Creating a Customized Crosstabulation Table Template with No Legend

rows_header=RowsHeader cols_header=ColsHeader;
label = "Frequency Counts and Percentages";

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;

define header RowsHeader;
text _ROW_LABEL_ / _ROW_LABEL_ ^= '';
text _ROW_NAME_ ;
style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
space=0;
end;

define header ColsHeader;
text _COL_LABEL_ / _COL_LABEL_ ^= '';
text _COL_NAME_ ;
style=header {backgroundcolor=#BFCFFF color=#6078bf fontstyle=italic};
space=1;
end;

define header ControllingFor;
dynamic StratNum StrataVariableNames StrataVariableLabels;
text "Controlling for" StrataVariableNames / StratNum > 0;
style=header;
end;

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;

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;
cellvalue
    Frequency Expected Deviation
    CellChiSquare TotalPercent Percent
    RowPercent ColPercent CumColPercent;
    header TableOf ControllingFor;
    footer NoObs Missing;
end;

ods html file='MyCrosstabsTable.html' style=white;
title "City Government Form by Number of Meetings Scheduled";
footnote "Cellvalues are stacked in the following order:";
*Example 1: Creating a Customized Crosstabulation Table Template with No Legend*

```
footnote2  "Frequency";
footnote3  "Percent";
footnote4  "Row Percent";
footnote5  "Column Percent";
ods noproctitle;

proc freq;
  tables citygovt*robgrp / missprint;
run;
ods html close;
```

**Program Description**

**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
```

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.htmlblue;
    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.

```
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
;
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 whether 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 heading 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 whether a label is assigned to the variable. If there is no label, the next TEXT statement, which specifies the row name, is 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 heading 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 whether a label is assigned to the column variable. If there is no label, the next TEXT statement, which specifies the row name, is 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 CELLSYIELD 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 than 50, then the font color is red.

```plaintext
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="";
```
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.

```
Define cellvalue ColPercent;
  header="";
  label="Percent of Column Frequency";
  format=6.2 print=on;
end;
```

Specify which headers and footers 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

Output 12.1 Output Using Customized Crosstabulation Table Template

<table>
<thead>
<tr>
<th>City Government Form</th>
<th>Number of Meetings Scheduled</th>
<th>?</th>
<th>Not Known</th>
<th>100 or Less</th>
<th>101-200</th>
<th>201-300</th>
<th>Over 300</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Council Manager</td>
<td></td>
<td>0</td>
<td>0</td>
<td>47</td>
<td>63</td>
<td>49</td>
<td>52</td>
<td>211</td>
</tr>
<tr>
<td>Commission</td>
<td></td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>Mayor Council</td>
<td></td>
<td>1</td>
<td>0</td>
<td>31</td>
<td>37</td>
<td>27</td>
<td>55</td>
<td>150</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>84</td>
<td>107</td>
<td>79</td>
<td>112</td>
<td>382</td>
</tr>
</tbody>
</table>

Frequency Missing = 12

Cells values are stacked in the following order:

- Frequency
- Percent
- Row Percent
- Column Percent
Example 2: Creating a Crosstabulation Table Template with a Customized Legend

Features:

- `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

Output 12.2  Output Using Default Crosstabulation Table
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 cell values

Program

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'
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 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
;
ods path (prepend) work.templat(update);
proc template;
  define crosstabs Base.Freq.CrossTabFreqs;
    notes "Crosstabulation table with legend";
    rows_header=RowsHeader cols_header=ColsHeader;
    label = "Frequency Counts and Percentages";
    grand_total_style=data {fontweight=bold};
  define header ControllingFor;
    dynamic StratNum StrataVariableNames StrataVariableLabels;
    text "Controlling for" StrataVariableNames / StratNum > 0;
    style=header;
  end;
  define header RowsHeader;
    text _ROW_LABEL_ / _ROW_LABEL_ ^= '';
    text _ROW_NAME_;
    space=0;
    style=header;
    cindent='';
  end;
  define header ColsHeader;
    text _COL_LABEL_ / _COL_LABEL_ ^= '';
    text _COL_NAME_;
    space=1;
    style=header;

Example 2: Creating a Crosstabulation Table Template with a Customized Legend

```
cindent=';';
end;

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_ ^=''
    & _COL_LABEL_ ^= '';
  text _ROW_NAME_ " by " _COL_NAME_ / NoTitle = 0;
  style=header;
end;

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;

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;

header ControllingFor;
footer TableOf NoObs Missing;
cellvalue
  Frequency Expected Deviation
  CellChiSquare TotalPercent Percent
  RowPercent ColPercent CumColPercent;
end;
runch;
title "City Government Form by Number of Meetings Scheduled";
ods html file='MyCrosstabsTableLegend.html' style=ocean;
proc freq;
  tables citygovt*robgrp / missprint;
Program Description

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
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 heading, a column heading, 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 whether 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 open parenthesis.

```
define header RowsHeader;
  text _ROW_LABEL_ / _ROW_LABEL_ ^= '';
  text _ROW_NAME_;
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 whether 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 open parenthesis.

```plaintext
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".

```plaintext
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_ ^= '' & _COL_NAME_ ^= ''; 
   text "Table " StratNum 3. " of " _ROW_NAME_ " by " _COL_LABEL_ / NoTitle= 0 & StratNum > 0 & _COL_LABEL_ ^= '';
   text "Table " StratNum 3. " of " _ROW_LABEL_ " by " _COL_LABEL_ / NoTitle= 0 & _ROW_LABEL_ ^= '' & _COL_LABEL_ ^= '';
   text _ROW_LABEL_ " by " _COL_LABEL_ / NoTitle = 0 & _ROW_LABEL_ ^= '';
   text _ROW_NAME_ " by " _COL_NAME_ / NoTitle = 0 & _ROW_LABEL_ ^= '';
   text _ROW_NAME_ " by " _COL_LABEL_ / NoTitle = 0 & _ROW_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.

```plaintext
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.

```plaintext
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 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.
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 with 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

Output 12.3 Output Using Customized Crosstabulation Table Template

### City Government Form by Number of Meetings Scheduled

#### The FREQ Procedure

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Row Percent</th>
<th>Column Percent</th>
<th>City Government Form</th>
<th>Number of Meetings Scheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Known</td>
<td>100 or Less</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Applicable</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Council Manager</td>
<td>0 47 63 49 52 211</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12.30 16.49 12.03 13.61 55.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.27 29.06 23.22 24.64 19.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>55.95 56.08 62.07 46.43 39.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Commission</td>
<td>0 4 7 3 5 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.57 1.03 0.79 1.31 5.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.57 33.33 14.29 23.01 100.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mayor Council</td>
<td>0 38 37 27 55 210</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>0.12 9.69 7.07 14.40 39.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.67 24.67 18.00 36.67 69.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>36.90 34.58 34.18 49.14 100.00</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>84</td>
<td>107 79 112</td>
<td>382</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21.99 20.01 20.68 29.32 100.00</td>
</tr>
</tbody>
</table>

City Government Form by Number of Meetings Scheduled

Frequency Missing = 12
Example 3: Adding Custom Formats to Cellvalues

Features:
- EDIT statement
- Other ODS features
  - ODS HTML statement
  - ODS PATH statement

Details

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: Creating a Crosstabulation Table Template with a Customized Legend” on page 402 by changing the formats of several cellvalues. In “Example 2: Creating a
Crosstabulation Table Template with a Customized Legend” on page 402, 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**

```sas
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
```

Chapter 12 / TEMPLATE Procedure: Creating Crosstabulation Table Templates
Example 3: Adding Custom Formats to Cellvalues

Program Description

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'
  run;
ods html file="userinfo.html" style=ocean;
title "Applying Custom Formats to Cellvalues";
proc freq;
  tables citygovt*robgrp / missprint;
run;
ods html close;
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: Creating a
Crosstabulation Table Template with a Customized Legend” on page 402.

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 with 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 12.5  Crosstabulation Output with Custom Formats Applied to Cellvalues

<table>
<thead>
<tr>
<th>City Government Form</th>
<th>Number of Meetings Scheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not Known</td>
</tr>
<tr>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>Not Applicable</td>
<td>0</td>
</tr>
<tr>
<td>Council Manager</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>12.304</td>
</tr>
<tr>
<td></td>
<td>22.275</td>
</tr>
<tr>
<td></td>
<td>55.952</td>
</tr>
<tr>
<td>Commission</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>28.571</td>
</tr>
<tr>
<td></td>
<td>7.143</td>
</tr>
<tr>
<td>Mayor Council</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>20.667</td>
</tr>
<tr>
<td></td>
<td>36.905</td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>84</td>
</tr>
</tbody>
</table>

City Government Form by Number of Meetings Scheduled

Frequency Missing = 12
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 and 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 perform the following steps:

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 that you need to analyze your data. For example, you can create the following model fit plot with these statements:
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.)
The following two graphics are just examples of what you can do with ODS graphics.

**Figure 13.2** PROC SGSCATTER (SAS) with LISTING Style

 Scatterplot Matrix for Iris Data

<table>
<thead>
<tr>
<th>Sepal Length (mm)</th>
<th>Petal Length (mm)</th>
<th>Sepal Width (mm)</th>
<th>Petal Width (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Species</td>
<td>S. setosa</td>
<td>V. versicolor</td>
<td>V. virginica</td>
</tr>
</tbody>
</table>

Iris Species: O: S. setosa, △: V. versicolor, ✗: V. virginica
For complete documentation on the syntax and usage of the Graph Template Language, see the following documentation: *SAS Graph Template Language: Reference* and *SAS Graph Template Language: User’s Guide*.

**Syntax**

```sas
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 Template Language: Reference*.

Creating statistical graphics with ODS:
For usage information about PROC TEMPLATE and the Graph Template Language, see *SAS 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 11, “TEMPLATE Procedure,” on page 333.

Modifying an existing style or creating your own style:
For reference information about the style template statements in PROC TEMPLATE, see Chapter 14, “TEMPLATE Procedure,” on page 425.
TEMPLATE Procedure: Creating a Style Template

Overview: TEMPLATE Procedure: Creating a Style Template
Using the TEMPLATE Procedure to Create a Style
Default Style for HTML
Customized Version of the HTML Style

Concepts: TEMPLATE Procedure: Creating a Style Template
Terminology
Viewing the Contents of a Style
Working with Styles
ODS Styles with Graphical Style Information
Understanding Styles, Style Elements, and Style Attributes
Understanding Inheritance
Understanding Style References
Using the FROM Option
Inheritance Compatibility across Versions

Syntax: TEMPLATE Procedure: Creating a Style Template
PROC TEMPLATE Statement
DEFINE STYLE Statement
CLASS Statement
EDIT Statement
END Statement
IMPORT Statement
NOTES Statement
PARENT= Statement
REPLACE Statement
STYLE Statement

Usage: TEMPLATE Procedure: Creating a Style Template
About Style Attributes

Examples: TEMPLATE Procedure: Creating a Style Template
Example 1: Creating a Stand-Alone Style
Example 2: Using User-Defined Attributes
Example 3: Modifying the Default Style with the CLASS Statement
Example 4: Defining a Table and Graph Style
Example 5: Defining Multiple Style Elements in One STYLE Statement
Example 6: Importing a CSS File
Example 7: Table Header and Footer Border Formatting
Example 8: Enhancing Titles and Footnotes in PDF Output
Example 9: Customizing Graphic and Tabular Titles in PDF Output
Overview: TEMPLATE Procedure: Creating a Style Template

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 specifies. However, you can also customize the appearance of the output by using the DEFINE STYLE statement in the TEMPLATE procedure.

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. The first output that follows shows the HTML output from PROC PRINT using the default style. The second output that follows shows the same HTML output from PROC PRINT with a customized style. The default style for HTML output is Styles.HTMLBlue.
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 types 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:

- the background of the contents file
- the background of the contents title
- the name of the table of contents (Contents instead of Table of Contents)

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 colors of the borders
- the background colors
Chapter 14 / TEMPLATE Procedure: Creating a Style Template

Concepts: TEMPLATE Procedure: Creating a Style Template

Terminology

For more definitions of terms used in this section, see “Terminology: TEMPLATE Procedure” on page 322.

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.
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.htmlblue;
     run;
     ```
  2. View any of the SAS styles by specifying **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** item store in the template store Sashelp.Tmplmst, along with the other styles that are supplied by SAS. You can view the styles from the Templates 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 14.1 Recommended Styles for ODS Destinations

<table>
<thead>
<tr>
<th>Destination</th>
<th>Recommended Styles</th>
<th>Default Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPUB</td>
<td>Daisy¹</td>
<td>Daisy</td>
</tr>
<tr>
<td></td>
<td>Moonflower</td>
<td></td>
</tr>
<tr>
<td>Printer family of statements</td>
<td>FancyPrinter</td>
<td>Pearl</td>
</tr>
<tr>
<td></td>
<td>FestivalPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrayscalePrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MeadowPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MonoChromePrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monospace</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NormalPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pearl</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Printer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sapphire</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SasDocPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SeasidePrinter</td>
<td></td>
</tr>
<tr>
<td>RTF</td>
<td>RTF</td>
<td>RTF</td>
</tr>
<tr>
<td>TAGSETS.RTF</td>
<td>RTF</td>
<td>RTF</td>
</tr>
<tr>
<td>ODS destination for PowerPoint</td>
<td>PowerPointDark</td>
<td>PowerPointLight</td>
</tr>
<tr>
<td></td>
<td>PowerPointLight</td>
<td></td>
</tr>
<tr>
<td>SASREPORT for Enterprise Guide</td>
<td>EGDefault</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td></td>
<td>HTMLBlue</td>
<td></td>
</tr>
<tr>
<td>SASREPORT for Web Report Studio</td>
<td>Normal</td>
<td>Plateau</td>
</tr>
<tr>
<td></td>
<td>Plateau</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seaside</td>
<td></td>
</tr>
<tr>
<td>LISTING</td>
<td>Listing</td>
<td>Listing</td>
</tr>
<tr>
<td>HTML</td>
<td>Minimal</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td></td>
<td>EGDefault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seaside</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HTMLBlue</td>
<td></td>
</tr>
</tbody>
</table>

¹ Daisy is recommended for EPUB to avoid any special characters that might cause problems.
<table>
<thead>
<tr>
<th>Destination</th>
<th>Recommended Styles</th>
<th>Default Style</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BarrettsBlue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BlockPrint</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dove</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dtree</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Festival</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gantt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Harvest</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HighContrast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal1a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal2a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal3a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Meadow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Netdraw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NoFontDefault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ocean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Plateau</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Raven</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SasWeb</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seaside</td>
<td></td>
</tr>
<tr>
<td></td>
<td>StatDoc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Statistical</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAGSETS.EXCELXP</th>
<th>Default</th>
<th>Default</th>
</tr>
</thead>
</table>

1 The Moonflower style for ODS EPUB is designed for nighttime or low-light reading.

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: Modifying the Default Style with the CLASS Statement" on page 474 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 Chapter 20, “Style Elements,” on page 797.
ODS Styles with Graphical Style Information

SAS provides ODS styles that incorporate graphical style information. These styles use 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: Defining a Table and Graph Style” on page 481, 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 gray-scale 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 Tables” on page 828 for a complete listing of style attributes. For a complete list of style elements see Chapter 20, “Style Elements,” on page 797.

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: Defining a Table and Graph Style” on page 481 describes how a defined ODS style was generated.

See “Viewing the Contents of a Style” on page 429 for information about viewing the code for the ODS styles that are delivered with SAS.

Understanding Styles, Style Elements, and Style Attributes

Understanding Styles, Style Elements, and Style Attributes

The appearance of SAS output is controlled by ODS style templates (ODS styles). ODS styles are produced from compiled STYLE templates written in PROC
TEMPLATE style syntax. An ODS style template is a collection of style elements that provides specific visual attributes for your SAS output.

- A style element is a named collection of style attributes that apply to a particular part of the output. Each area of ODS output has a style element name that is associated with it. The style element name specifies where the style attributes are applied. 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.

- A style attribute is a visual property, such as color, font properties, and line characteristics, that is defined in ODS with a reserved name and value. Style attributes are collectively referenced by a style element within a style template. Each style attribute specifies a value for one aspect of the presentation. For example, the BACKGROUNDCOLOR= 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 font or an italic font.

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

Available styles are in the SASHELP.TMPLMST item store. In SAS Enterprise Guide, the list of style sheets is shown by the Style Wizard. In batch mode or SAS Studio, you can display the list of available style templates by using the LIST statement in PROC TEMPLATE:

```
proc template;
    list styles / store=sashelp.tmplmst;
run;
```

For complete information about viewing ODS styles, see “Viewing ODS Styles Supplied by SAS” on page 727.

By default, HTML 4 output uses the HTMLBlue style template and HTML 5 output uses the HTMLEncore style template. To help you become familiar with styles, style elements, and style attributes, look at the relationship between them.

You can use the SOURCE statement in PROC TEMPLATE to display the structure of a style template. The following code prints the structure of the HTMLBlue style template to the SAS log:

```
proc template;
    source styles.HTMLBlue;
run;
```

The following figure illustrates the structure of a style. The figure shows the relationship between the style, the style elements, and the style attributes.
The following list corresponds to the numbered items in the preceding figure:

1 Styles.HTMLBlue is the style. Styles describe how to display presentation aspects (color, font, font size, and so on) of the SAS output. A style determines the overall appearance of the ODS documents that use it. The default style for HTML output is HtmlBlue. Each style consists of style elements.
You can create new styles with the "DEFINE STYLE Statement" on page 448. New styles can be created independently or from an existing style. You can use "PARENT= Statement" on page 454 to create a new style from an existing style. For complete documentation about ODS styles, see Chapter 19, "Style Templates," on page 727.

Header and Footer are examples of 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 styles and consist of one or more style attributes. Style elements can be user-defined or supplied by SAS. User-defined style elements can be created by the "STYLE Statement" on page 455.

Note: For a list of the default style elements used for HTML and markup languages and their inheritance, see Chapter 20, "Style Elements," on page 797.

BORDERCOLOR=, BACKGROUNDCOLOR=, and COLOR= are examples of style attributes. Style attributes specify a value for one aspect of the area of the output that its style element applies to. For example, the COLOR= attribute specifies the value "#112277" for the font color. For a list of style attributes supplied by SAS, see Chapter 21, "Style Attributes," on page 827.

Style attributes can be referenced with style references. See "style-reference" on page 894 for more information about style references.

The following table shows commonly used style attributes that you can set with the STYLE= option in PROC PRINT, PROC TABULATE, and PROC REPORT. Most of these attributes apply to parts of the table other than cells (for example, table borders and the lines between columns and rows). Note that not all attributes are valid in all destinations. For more information about these style attributes, their valid values, and their applicable destinations, see "Style Attributes Tables" on page 828.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</th>
<th>PROC TABULATE STATEMENT TABLE</th>
<th>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</th>
<th>PROC PRINT: all locations other than TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BACKGROUNDCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BACKGROUNDIMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Attribute</td>
<td>PROC REPORT STATEMENT REPORT Area</td>
<td>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</td>
<td>PROC TABULATE STATEMENT TABLE</td>
<td>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>BORDERBOTTOMSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTCOLOR=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLOR=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLORDARK=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLORLIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTCOLOR=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPCOLOR=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CELLPADDING=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CELLSPACING=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Attribute</td>
<td>PROC REPORT STATEMENT REPORT Area</td>
<td>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</td>
<td>PROC TABULATE STATEMENT TABLE</td>
<td>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------------------------------</td>
<td>-------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>CELLWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CLASS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>COLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FLYOVER=</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FONT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FONTFAMILY=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FONT SIZE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FONT STYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FONT WEIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FONT WIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FRAME=</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HREFTARGET=</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTML STYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>NOBREAK SPACE=²</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTPUT WIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POST HTML=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POST IMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POST TEXT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PRE HTML=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PRE IMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PRE TEXT=³</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PROTECT SPECIAL CHAR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
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 14.1  Original Code for Creating Style2**

```latex
define style style2;
  style fonts /
    "docfont" = ("Arial", 3)
    "tablefont" = ("Times", 2);
  style output /
    cellpadding = 5
    borderspacing = 0
    font = fonts("docfont");
  style table from output /
    borderspacing = 2
    font = fonts("tablefont");
  style header /
    backgroundcolor=white
    color=blue
    fontfamily="arial, helvetica"
    fontweight=bold;
end;
```

**Example Code 14.2  Expanded Version of Style2**

```latex
define style style2;
  parent = style1;
  style fonts from fonts /
    "docfont" = ("Helvetica", 3);
  style table from table /
    borderspacing = 4;
  style header /
    fontstyle=roman
    fontsize=5;
end;
```

The Style2 style from the previous code could also be written this way:

**Example Code 14.2  Expanded Version of Style2**

```latex
define style style2;
  style fonts/
    "docfont" = ("Helvetica", 3)
    "tablefont" = ("Times", 2);
  style output /
    cellpadding = 5
    borderspacing = 0
    font = fonts("docfont");
  style table from output /
    borderspacing=4
    font = fonts("tablefont");
```
Inheritance between Style Elements

The FROM option in 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 Example Code 14.3 on page 439 and the Example Code 14.4 on page 439 the Table style element, which is created with the style table from output / ... statement, ends up with the following style attributes:

- cellpadding= 5
- borderspacing= 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:

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
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 14.3  Program with Unresolved Style References**

```sas
define style style1;
  style fonts /
    "docfont" = {"Arial", 3}
    "tablefont" = {"Times", 2};
  style output /
    cellpadding = 5
    borderspacing = 0
    font = fonts("docfont");
  style table from output /
    borderspacing = 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 /
    borderspacing = 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 14.4  Program with Unresolved Style References**

```sas
define style style1;
  style fonts /
    "docfont" = {"Arial", 3}
    "tablefont" = {"Times", 2};
  style output /
    cellpadding = 5
```
Using the FROM Option

The FROM option is used with a style element in order to inherit from another style element. If you omit the FROM option, then you can have an incomplete style element in the child style.

For example, in the following SAS program, 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 Figure 14.20 on page 444.

To correct this, in the following program, you can add the FROM option to the STYLE statement that creates the Colors style element in Concepts.Style2:

```sas
style colors from colors /
   "dark"=dark blue;
```

Note: In the following code, Concepts is a folder that is created in Templates ⇒ Sasuser.Templat. Style1 is a template that is created in the Concepts folder.
Example Code 14.5  Creating the Colors Style Element without the FROM Option

```sas
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” in SAS Output Delivery System: User’s Guide.
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 442.

Note: In the following code, Concepts is a folder that is created in Templates \(\Rightarrow\) Sasuser.Templat. Style1 is a template that is created in the Concepts folder.

```sas
proc template;
define style concepts.style1;
  style celldatasimple /
    fontfamily=arial
    backgroundcolor=very light vivid blue
    color=white;
  style celldataemphasis from celldatasimple /
    color=red1
    fontstyle=italic;
  style celldatalarge from celldataemphasis /
    fontweight=bold
    fontsize=5;
end;
run;

proc template;
define style concepts.style2;
  parent=concepts.style1;
  style celldataemphasis from celldataemphasis3 /
    backgroundcolor=yellow2;
  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 and beyond. This is because, when you change the value of the COLOR= attribute in CellDataEmphasis from red (1) to yellow (2), the change affects only style elements that inherit from CellDataEmphasis in Concepts.Style2. Within Concepts.Style2, there are no style elements that inherit from CellDataEmphasis (3). Therefore, only CellDataEmphasis in Concepts.Style2 has yellow text. Beginning with SAS 9.2, all style elements in parent style templates 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” in SAS Output Delivery System: User’s Guide.
Syntax: TEMPLATE Procedure: Creating a Style Template

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;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC TEMPLATE</td>
<td>Begin a PROC TEMPLATE template</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7</td>
</tr>
<tr>
<td>DEFINE STYLE</td>
<td>Create a style for any destination that supports the STYLE= option</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7</td>
</tr>
<tr>
<td>CLASS</td>
<td>Create a style element from a like-named style element</td>
<td>Ex. 3, Ex. 6</td>
</tr>
<tr>
<td>END</td>
<td>End the style</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7</td>
</tr>
<tr>
<td>EDIT</td>
<td>Edit a style</td>
<td></td>
</tr>
<tr>
<td>IMPORT</td>
<td>Import Cascading Style Sheet (CSS) information from a file into the style</td>
<td>Ex. 6</td>
</tr>
<tr>
<td>NOTES</td>
<td>Provide information about the style</td>
<td></td>
</tr>
<tr>
<td>PARENT=</td>
<td>Specify the style from which the current style inherits</td>
<td>Ex. 3, Ex. 4, Ex. 6, Ex. 7</td>
</tr>
<tr>
<td>STYLE</td>
<td>Create or modify one or more style elements</td>
<td>Ex. 1, Ex. 2, Ex. 4, Ex. 5, Ex. 7</td>
</tr>
</tbody>
</table>

PROC TEMPLATE Statement

Begins a PROC TEMPLATE template.
Syntax

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)>;
  IMPORT style-specification <media-type-1 <, media-type-2> ...>;
  STYLE style-element-name(s) <FROM style-element-name | _SELF_ > <"text">
    </style-attribute-specification(s)>;
END;
END;

DEFINE STYLE Statement

Creates a style for any destination that supports the STYLE= option. The DEFINE STYLE statement begins a DEFINE STYLE statement block.

Requirement: An END statement must be the last statement in the template.

Supports: The DEFINE STYLE statement supports the following statements: "CLASS Statement" on page 449, "IMPORT Statement" on page 453, "NOTES Statement" on page 383, "PARENT= Statement" on page 454, and "STYLE Statement" on page 455.

Examples: "Example 1: Creating a Stand-Alone Style" on page 459
"Creating a Cover Page with the LAYOUT_ABSOLUTE Method" in SAS Output Delivery System: Advanced Topics
"Example 1: Adding Text to Multiple Destinations " on page 70
"Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576
"Example 7: Table Header and Footer Border Formatting" on page 586
"Example 1: Creating a Customized Crosstabulation Table Template with No Legend" on page 390
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.

To view the Base.Template.Style 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 Base to view the contents of that directory.
4. Double-click Template to view the contents of that directory.

**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.

### Restriction
If the PARENT= statement is specified, then PARENT= must refer to a style other than Base.Template.Style.

### Interaction
The Base.Template.Style master template attributes are overridden by other style templates.

### Tip
To view an existing style to base your own Base.Template.Style on, see “Viewing the Contents of a Style” on page 429.

Optional Argument

**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.
Restriction: The CLASS statement must be used within a DEFINE STYLE template block.

Example: The following statements are equivalent:

```plaintext
class fonts;
style fonts from fonts;
style fonts from _self_;
```

Examples: "Example 3: Modifying the Default Style with the CLASS Statement" on page 474
"Example 6: Importing a CSS File" on page 493

Syntax

```plaintext
CLASS style-element-name(s) "<text"> </style-attribute-specification(s)>;
```

Required Argument

**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** For a complete description of **style-element-name**, see "**style-element-name**" on page 456 in the STYLE statement.

For a list of style elements, see Chapter 20, "Style Elements," on page 797.

Optional Arguments

**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:

```plaintext
style-attribute-name=< | >style-attribute-value
```

**style-attribute-name**

is the name of an attribute that is listed in “Style Attributes Tables” on page 828, or it is the name of a user-defined style attribute.

**Tip** 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 Tables” on page 828, then do not enclose the name in quotation marks.

**style-attribute-value**

assigns the value to the attribute. If an attribute from the table in “Style Attributes Tables” on page 828 is specified, then specify the type of value that the attribute expects.

For more information about style-attribute values, see Chapter 21, "Style Attributes," on page 827.

| 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.

Tips
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.

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.

EDIT Statement
Edits an existing template. The EDIT statement replaces the DEFINE statement in a template block when editing. You can use the EDIT statement in place of any DEFINE statement.

Restriction: If you edit a template that is a link, the link is broken and a separate template is created.
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.

Example: “Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551

Syntax
EDIT template-path-1 <AS template-path-2 > <STORE=libref.template-store>>;

template-statements;
END;

Required Argument
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 76.)
Optional Arguments

**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.

**template-statements**

*template-statements* are any statements or attributes that are valid between the DEFINE statement and the END statement.

Editing an Existing Template

When you use the EDIT statement, the following occurs:

- 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 344.) It opens a copy of the first template path that it finds in a template store that has Read access.

- 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.

END Statement

Ends the style.

**Requirement:** An END statement must be the last statement in the template.

**Syntax**

```plaintext
END;
```
IMPORT Statement

Imports Cascading Style Sheet (CSS) information from a file into the style.

Restriction: The IMPORT statement must be used within a DEFINE STYLE template block.

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.

Example: “Example 6: Importing a CSS File” on page 493

Syntax

IMPORT file-specification <media-type-1 <, media-type-2 …>>

Required Argument

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 DATA Step Statements: Reference.

“URL” is a URL to an external CSS file.

Requirement You must enclose external-file in quotation marks.

Optional Argument

media-type 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.

**Restriction:**
The NOTES statement must be used within a DEFINE STYLE template block.

**Tip:**
The NOTES statement becomes part of the compiled style template, which you can view with the SOURCE statement, whereas SAS comments do not.

#### Syntax

```plaintext
NOTES "text";
```

**Required Argument**

*"text"*

provides information about the style.

---

### PARENT= Statement

Specifies the style from which the current style inherits.

**Restriction:**
The PARENT= statement must be used within a DEFINE STYLE template block.

**Examples:**

"Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390
"Creating a Cover Page with the LAYOUT_ABSOLUTE Method” in SAS Output Delivery System: Advanced Topics
"Example 1: Adding Text to Multiple Destinations " on page 70
"Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576
"Example 7: Table Header and Footer Border Formatting” on page 586
Syntax

PARENT=style-path;

Required Argument

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.Journal
- Styles.Grasyscaleprinter
- Styles.Statistical
- Styles.Minimal
- Styles.Pearl
- 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 429.

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 or the CLASS statement to create and modify style elements.

STYLE Statement

Creates or modifies one or more style elements.

Restriction: The STYLE statement must be used within a DEFINE STYLE template block.

Example: “Example 1: Creating a Stand-Alone Style” on page 459
Syntax

**STYLE** *style-element-name(s)*

```xml
<FROM existing-style-element-name | _SELF_ >"text">
</style-attribute-specification(s)>;
```

Required Argument

*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.

Style element inheritance follows these general guidelines:

- 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.
- 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.
- 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.
- If there are multiple identical 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.

**Requirement** Style elements must be separated by commas.

**See** Chapter 20, “Style Elements,” on page 797 for a list of style elements

**Example** The following STYLE statement uses a style element list:

```xml
style data, data1, dataempty from _self_ /
    color = red
    backgroundcolor = black;
```

That statement is equivalent to specifying the following STYLE statements together:

```xml
style data from data /
    color = red
    backgroundcolor = black;
style data1 from data1/    color = red
    backgroundcolor = black;
style dataempty from dataempty /
    color = red
    backgroundcolor = black;
```

**Example** “Example 5: Defining Multiple Style Elements in One STYLE Statement” on page 488
Optional Arguments

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. Style inheritance using the FROM option follows these general guidelines:

- 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.
- 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.
- 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.
- 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 Data1 be created from the style element Data2, and that the COLOR=BLACK style attribute be added.

```bash
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.

See

Chapter 20, “Style Elements,” on page 797 for a list of style elements

Example

The following STYLE statement uses the FROM _SELF_ option:

```bash
style data, data1, dataempty from _self_/ 
  color = red backgroundcolor = black;
```

That statement is equivalent to specifying the following STYLE statements together:

```bash
style data from data / 
  color = red 
  backgroundcolor = black;

style data1 from datal / 
  color = red 
  backgroundcolor = black;
```
style dataempty from dataempty /
  color = red
  backgroundcolor = black

**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 Tables” on page 828, or it is the name of a user-defined style attribute.

**Tip** 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 Tables” on page 828, then do not enclose the name in quotation marks.

**style-attribute-value**
assigns the value to the attribute. If an attribute from the table in “Style Attributes Tables” on page 828 is specified, then specify the type of value that the attribute expects.

For more information about style-attribute values, see Chapter 21, “Style Attributes,” on page 827.

| 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.

**Tips** 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.

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.
Usage: TEMPLATED Procedure: Creating a Style Template

About Style Attributes

Overview

Style attributes influence the characteristics of individual cells, tables, documents, graphs, and HTML frames. Style attributes exist within style elements and are specified by the STYLE statement on page 455 or the CLASS statement on page 449. 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 432. For information about using style attributes with ODS Statistical Graphics, see the chapter on controlling the appearance of your graphics in SAS Graph Template Language: User’s Guide.

Style attributes can be supplied by SAS or user-defined. Style attributes can be referenced with a style reference. See “Understanding Style References” on page 440 and “style-reference” on page 894 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 429.

For a list of the values that style attributes can specify, see Chapter 21, “Style Attributes,” on page 827. For a list of style elements that you can specify style attributes in, see Chapter 20, “Style Elements,” on page 797.

Examples: TEMPLATED Procedure: Creating a Style Template

Example 1: Creating a Stand-Alone Style

Features:

BACKGROUND_COLOR= in the STYLE statement
BORDER_WIDTH= in the STYLE statement
BORDER_SPACING= in the STYLE statement
FONT_FAMILY= in the STYLE statement
FONT_SIZE= in the STYLE statement
Details

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: Modifying the Default Style with the CLASS Statement” on page 474). 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 Chapter 20, “Style Elements,” on page 797.

Program

```proc template;
  define style newstyle;
    style cellcontents /
      fontfamily="arial, helvetica"
      fontweight=medium
      backgroundcolor=blue
      fontstyle=roman
      fontsize=5
      color=white;

    style header /
      backgroundcolor=very light blue
      fontfamily="arial, helvetica"
      fontweight=medium
      fontstyle=roman
      fontsize=5
      color=white;
```
Example 1: Creating a Stand-Alone Style

```plaintext
style systemtitle /
  fontfamily="arial, helvetica"
  fontweight=medium
  backgroundcolor=white
  fontstyle=italic
  fontsize=6
  color=red;
style footer from systemtitle /
  fontsize=3;
style table /
  borderspacing=5
  borderwidth=10;
end;
run;
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;

data _null_
set grain_production;
where type in ("Rice", "Corn") and year=1996;
file print ods=(
    template="table1"
    columns=(
      char_var=country(generic=on format=$cntry.
      dynamic=(colhd="Country")))
```
Program Description

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 consists of the style attributes that appear in 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;
  
  style header /
    backgroundcolor=very light blue
    fontfamily="arial, helvetica"
    fontweight=medium
    fontstyle=roman
    fontsize=5
    color=white;

  style systemtitle /
    fontfamily="arial, helvetica"
    fontweight=medium
    backgroundcolor=white
    fontstyle=italic
    fontsize=6
    color=red;
run;
```

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.

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.
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 /
    borderspacing=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 hardcoding 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.

```
declare charVar;
    generic;
    blank_dups=on;
    header='Colhd';
    style=cellcontents;
end;
```

Example 1: Creating a Stand-Alone Style
**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.

```plaintext
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.

```plaintext
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.

```plaintext
end;
run;
```

**Create a file reference for the output.** The current working directory is specified in this example.

```plaintext
filename odsout ".";
ods html close;
```

**Create HTML output and specify the location for storing the HTML output.**

**Specify the style to use for the output.** The HTML destination is open by default. However, to specify a style, you must use the ODS HTML statement with the STYLE= open specified. The STYLE= option tells ODS to use NewStyle as the style when it formats the output.

```plaintext
ods html path=odsout file="newstyle-body.htm" style=newstyle;
```

**Specify the titles for the report.** The TITLE statements provide two titles for the output.

```plaintext
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.

```plaintext
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. The TEMPLATE= suboption tells ODS to use the table template named Table1, which was previously created with PROC TEMPLATE.

For more information about using the DATA step with ODS, see “Using ODS with the DATA Step” in SAS Output Delivery System: User’s Guide.

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 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 HTML statement opens the HTML destination to return ODS to its default setup.

    ods html close;
    ods html;

HTML Output: Specifying Colors and Fonts with User-Defined Attributes

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.
Example 2: Using User-Defined Attributes

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10,035</td>
</tr>
<tr>
<td></td>
<td>Corn</td>
<td>31,975</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190,100</td>
</tr>
<tr>
<td></td>
<td>Corn</td>
<td>119,350</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>120,012</td>
</tr>
<tr>
<td></td>
<td>Corn</td>
<td>8,660</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>51,165</td>
</tr>
<tr>
<td></td>
<td>Corn</td>
<td>8,925</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7,771</td>
</tr>
<tr>
<td></td>
<td>Corn</td>
<td>236,064</td>
</tr>
</tbody>
</table>

Prepared on 11FEB2011
Program 1: Details

This example creates a style that is equivalent to the style that “Example 1: Creating a Stand-Alone Style” on page 459 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 a Custom Style with User-Defined Style Attributes

```
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;

       style colors /
           "light"=white
           "medium"=cxaaaaff
           "dark"=cx0000ff
           "bright"=red;

       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 /
           borderspacing=5
           borderwidth=10;

   end;
run;

define table table1;
   mvar sysdate9;
dynamic colhd;
classlevels=on;
define column char_var;
    generic=on;
    blank_dups=on;
```
Program Description

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 consists 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";
run;
ods html close;
ods html;
Create the style element Colors. This STYLE statement defines the style element Colors. This style element consists 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 /
  borderspacing=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 DEFINE TABLE statement creates a new table template called Table1.

   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 hardcoding 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;
```

**Create HTML output and specify the location for storing the HTML output. Specify the style to use for the output.** The HTML destination is open by default.
However, to specify a style, you must use the ODS HTML statement with the STYLE= open specified. 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. The TEMPLATE= suboption tells ODS to use the table template named Table1, which was previously created with PROC TEMPLATE.

For more information about using the DATA step with ODS, see “Using ODS with the DATA Step” in SAS Output Delivery System: User’s Guide.

```
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;
```
**Close the HTML destination.** The ODS HTML statement closes the HTML destination and all the files that are associated with it. The ODS HTML statement opens the HTML destination to return ODS to its default setup.

```plaintext
ods html close;
ods html;
```

**HTML Output**

This HTML output is identical to the output in the section “HTML Output: Specifying Colors and Fonts with User-Defined Attributes” on page 465, 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.

**Output 14.2 HTML Output**

![Leading Grain Producers in 1996](image)

**Program 2: Details**

In the program "Example 1: Creating a Stand-Alone Style" on page 459, 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 the Color Scheme

```
style colors /
  "light"=white
  "medium"=cxaaaaff
  "dark"=cx0000ff
  "bright"=red;
style colors /
  "light"=white
  "medium"=pink
  "dark"=red
  "bright"=red;

"cellfont"=("arial, helvetica", 4, medium roman)
"cellfont"=("courier, arial, helvetica", 4, medium roman)
```

Program Description

**To make the color scheme change, change only this section of code:**

The following is the original portion on code from "Program 1: Creating a Custom Style with User-Defined Style Attributes" on page 467.

```
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)
```

**HTML Output: Changing Colors and Fonts of User-Defined Attributes**

This HTML output shows the results of running the same program with these changes.

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.
Example 3: Modifying the Default Style with the CLASS Statement

Features:
- DEFINE STYLE statement
- User-defined attributes
- BACKGROUNDCOLOR= style attribute
- BORDERWIDTH= style attribute
- CELLPADDING= style attribute
- BORDERSPACING= style attribute
- COLOR= style attribute
- FONT= style attribute
- FONTSTYLE= style attribute
- FRAME= style attribute
- POSTHTML= style attribute
- RULES= style attribute
Details

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 BY line, 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 border spacing, 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, 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

```sas
proc template;
   define style customdefault;
      parent=styles.htmlblue;
      class contents /
         background=cxfffffc;

      class contenttitle /
         background=cxfffffc;

      class data /
         background=cxcxxxxc;
```
style IndexProcName from Index /

  backgroundcolor = cxffffcc;

class colors /
  'link2' = cx0000FF
  'link1' = cx800080
  'docbg' = cx99ccff
  'contentbg' = cxFAFBFE
  'systitlebg' = cx99ccff
  'titlebg' = cxFAFBFE
  'proctitlebg' = cxFAFBFE
  'headerbg' = cxEDF2F9
  'captionbg' = cxFAFBFE
  'captionfg' = cx112277
  'bylinebg' = cxffffcc
  'notebg' = cxFAFBFE
  'tablebg' = cxFAFBFE
  'batchbg' = cxFAFBFE
  'systitlefg' = cx112277
  'titlefg' = cx112277
  'proctitlefg' = cx112277
  'bylinefg' = cx112277
  'notefg' = cx112277;

class Header /
  bordercolor = cxEDF2F9
  backgroundcolor = cxEDF2F9
  color = cx112277;

class text /
  "prefix1" = "PROC "
  "suffix1" = ":"
  "Content Title" = "Contents"
  "Pages Title" = "Pages"
;

end;
run;

filename odsout ".";
ods html close;

ods html path=odsout body="customdefaultstyle-body.htm"
  contents="customdefaultstyle-content.htm"
  frame="customdefaultstyle-frame.htm"
  style=customdefault;

title "MSRP by Make and Model";
title2 "(6 Cylinders Only)*";

proc print data=sashelp.cars noobs;
  var make model cylinders MSRP;
  format MSRP dollar10.2;
  by make;
  where cylinders=6;
run;
ods html close;
ods html;
Program Description

Create the style CustomDefault. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called CustomDefault.

```latex
proc template;
define style customdefault;
```

Specify the parent style from which the CustomDefault style inherits its attributes. The PARENT= attribute specifies Styles.HTMLBlue 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.

```latex
parent=styles.htmlblue;
```

Customize the contents style elements and the data cells. By changing the BACKGROUND= style attribute in the style elements Contents, ContentTitle, and IndexProcName, the background of the contents becomes yellow.

```latex
class contents  /
    background=cxffffcc;

class contenttitle /
    background=cxffffcc;

class data  /
    background=cxcccccc;
style IndexProcName from Index /
    backgroundcolor = cxffffcc;
```

Change the attributes of the style element Colors This CLASS statement adds to the child style the style element Colors, which also exists in the parent style (HTMLBlue). The CLASS statement adds all of the style attributes that are in the original instance of the Colors style element to the new instance of Colors, except for those that are overridden by the new instance of Colors. By using the CLASS statement, you do not need to specify the FROM option. If you did not use the CLASS statement or the FROM option, then the attributes from the original instance of Colors would not be added to the new instance of Colors. The Colors style element in CustomDefault would contain only the style statements that it specifically specifies. All style elements that use the user-defined attributes that Colors defines (fgB2, fgB1, and so on) use the style attributes that are specified in Custom.Default, not the ones that are specified in Styles.HTMLBlue. 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 three of the user-defined style attributes: Docbg=, Systitlebg=, and Bylinebg=.

```latex
class colors  /
    'link2' = cx0000FF
    'link1' = cx800080
    'docbg' = cx99ccff
    'contentbg' = cxFAFBFE
    'systitlebg' = cx99ccff
    'titlebg' = cxFAFBFE
    'proctitlebg' = cxFAFBFE
    'headerbg' = cxEDF2F9
    'captionbg' = cxFAFBFE
    'captionfg' = cx112277
```
Change the style attributes in the style element 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.

In the default style, the background color for the BY line differs from the background color for the document, so it appears as a gray stripe in the default output. In this customized style, the stripe disappears because the background color for the BY line and the document are the same.

Customize the text used in parts of the output. 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.)

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".

Stop the creation of the customized style. The END statement ends the style. The RUN statement executes the PROC TEMPLATE step.

Create a file reference for the output. The current working directory is specified in this example.

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. 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 path=odsout 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 "MSRP by Make and Model";
title2 "(6 Cylinders Only)";
```

Create the PRINT procedure output. This PROC PRINT step is the same one that was used with the default style in the previous program.

```
proc print data=sashelp.cars noobs;
   var make model cylinders MSRP;
   format MSRP dollar10.2;
   by make;
   where cylinders=6;
run;
```

Close the HTML destination. The ODS HTML statement closes the HTML destination and all the files that are associated with it. The ODS HTML statement opens the HTML destination to return ODS to its default setup.

```
ods html close;
ods html;
```
## Output 14.4  HTML Output from PROC PRINT Using the Default Style

### Table of Contents

1. The Print Procedure
   - Make=Acura
     - Data Set SASHELP.CARS
   - Make=Audi
     - Data Set SASHELP.CARS

### Energy Expenditures for Each Region (millions of dollars)

**Make=Acura**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Cylinders</th>
<th>MSRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acura</td>
<td>MDX</td>
<td>6</td>
<td>$36,945.00</td>
</tr>
<tr>
<td>Acura</td>
<td>TL 4dr</td>
<td>6</td>
<td>$33,195.00</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>6</td>
<td>$43,755.00</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>6</td>
<td>$46,100.00</td>
</tr>
<tr>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>6</td>
<td>$89,765.00</td>
</tr>
</tbody>
</table>

**Make=Audi**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Cylinders</th>
<th>MSRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>6</td>
<td>$31,840.00</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>6</td>
<td>$33,430.00</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>6</td>
<td>$34,480.00</td>
</tr>
</tbody>
</table>

## Output 14.5  HTML Output from PROC PRINT with the Customized Style

### Contents

1. Print
   - Make=Acura
     - Data Set SASHELP.CARS
   - Make=Audi
     - Data Set SASHELP.CARS

### MSRP by Make and Model (6 Cylinders Only)

**Make=Acura**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Cylinders</th>
<th>MSRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acura</td>
<td>MDX</td>
<td>6</td>
<td>$36,945.00</td>
</tr>
<tr>
<td>Acura</td>
<td>TL 4dr</td>
<td>6</td>
<td>$33,195.00</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>6</td>
<td>$43,755.00</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>6</td>
<td>$46,100.00</td>
</tr>
<tr>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>6</td>
<td>$89,765.00</td>
</tr>
</tbody>
</table>

**Make=Audi**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Cylinders</th>
<th>MSRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>6</td>
<td>$31,840.00</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>6</td>
<td>$33,430.00</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>6</td>
<td>$34,480.00</td>
</tr>
</tbody>
</table>
Example 4: Defining a Table and Graph Style

Features:

DEFINE STYLE statement style attributes:

- User-defined attributes
- Style attribute: BACKGROUNDCOLOR=
- Style attribute: BORDERCOLORDARK=
- Style attribute: BORDERCOLORLIGHT=
- Style attribute: BORDERWIDTH=
- Style attribute: CELLPADDING=
- Style attribute: BORDERSPACING=
- Style attribute: DROPSHADOW=
- Style attribute: ENDCOLOR=
- Style attribute: FONT=
- Style attribute: COLOR=
- Style attribute: FRAME=
- Style attribute: GRADIENTDIRECTION=
- Style attribute: IMAGE=
- Style attribute: TEXTALIGN=
- Style attribute: style attribute:WIDTH=
- Style attribute: RULES=
- Style attribute: TRANSPARENCY=
- Style attribute: VERTICALALIGN=

DEFINE STYLE statement style elements:

- Style element: GraphAxisLines
- Style element: GraphBackground
- Style element: GraphBorderLines
- Style element: GraphCharts
- Style element: GraphLabelText
- Style element: GraphWalls

PARENT= statement

STYLE statement

Details

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

```
filename odsout ".*";
ods html close;
options nodate;
```
proc template;
  define style Styles.Science;

  parent = styles.default;

  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,Bold)
headingEmphasisFont" = ("Verdana, Verdana, Helvetica, sans-serif",14pt,Bold)
headingEmphasisFont" = ("Verdana, Verdana, Helvetica, sans-serif",8pt,Bold);

  style GraphFonts from _self_/ 
  "GraphValueFont" = ("Verdana",10pt)
"GraphLabelFont" = ("Verdana",14pt,Bold);

  style colors /
  "headerfgemph" = cx31035E
"headerbgemph" = cxFFFFFE
"headerfgstrong" = cx31035E
"headerbgstrong" = cxFFFFFE
"headerfg" = cx31035E
"headerbg" = cxFFFFFE
"datafgemph" = cx31035E
"databgemph" = cxDFECE1
"datafgstrong" = cx31035E
"databgstrong" = cxDFECE1
"datafg" = cx31035E
"databg" = cxDFECE1
"batchfg" = cx31035E
"batchbg" = cxDFECE1
"tablebg" = cx31035E
"tableborderdark" = cx909090
"tableborderlight" = cxDFECE1
"tableborder" = cxDFECE1
"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
Example 4: Defining a Table and Graph Style

<table>
<thead>
<tr>
<th>Example 4: Defining a Table and Graph Style</th>
<th>483</th>
</tr>
</thead>
</table>

```
"Confolderfg" = cx31035E
"Contitlefg" = cx31035E
"link2" = cx800080
"link1" = cx0000FF
"contentfg" = cx31035E
"contentbg" = cxDFECE1
"docfg" = cx31035E
"docbg" = cxDFECE1;

style GraphColors /
"gconramp3cend" = cxDD6060
"gconramp3cneutral" = cxFFFFFFF
"gconramp3cstart" = cx6497EB
"gramp3cend" = cxBED8D3
"gramp3cneutral" = cxFFFFFFF
"gramp3cstart" = cxAAAB6DF
"gconramp2cend" = cx6497EB
"gconramp2cstart" = cxFFFFFFF
"gramp2cend" = cx548287
"gramp2cstart" = cxFFFFFFF
"gtext" = CX31035E
"glabel" = CX31035E
"gborderlines" = CX31035E
"goutlines" = CX31035E
"ggrid" = CX31035E
"gaxis" = CX31035E
"gshadow" = CX707671
"glegend" = cxFFFFFFF
"gfloor" = CXDFECE1
"gwalls" = cxFFFFFFF
"gcdata12" = cxFF667F
"gcdata11" = cx5050CC
"gcdata10" = cxE100BF
"gcdata9" = cx007F00
"gcdata8" = cxB99600
"gcdata7" = cx7F7F7F
"gcdata6" = cx984EA3
"gcdata5" = cx4DAP4A
"gcdata4" = cxA65628
"gcdata3" = cxFF7F00
"gcdata2" = cx377DB8
"gcdata1" = cxE31A1C
"gdatal12" = CX4A5573
"gdatal11" = CXCFB1E2
"gdatal10" = CX3E8E29D
"gdatal9" = CX2952B1
"gdatal8" = CXAAB6DF
"gdatal7" = CX6771C2
"gdatal6" = CXBED8D3
"gdatal5" = CX8B65A3
"gdatal4" = CXBCD3AB
"gdatal3" = CX548287
"gdatal2" = CX7DC1C9
"gdatal1" = CX9580D5;

style Table from Output /
  cellpadding = 5
```
Program Description

Create a file reference for the output and set the SAS system options. The current working directory is specified in this example.

```
filename odsout ".";
ods html close;
options nodate;
```

Create the style Science. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style in the Styles item store 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 exists in the parent style. All style elements that use the user-defined attributes that Fonts 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 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)
  "BatchFixedFont" = ("Courier New", Courier, monospace",10pt)
  "FixedFont" = ("Courier New", Courier, monospace",10pt)
  "headingEmphasisFont" = ("Verdana, Verdana, Helvetica, sans-serif",14pt,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 are specifically specified in Science.

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
```
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.

```plaintext
style GraphColors /
  "gconramp3cend" = cxDD6060
  "gconramp3cneutral" = cxFFFFFFFF
  "gconramp3cstart" = cx6497EB
  "gramp3cend" = cxBED8D3
  "gramp3cneutral" = cxFFFFFFFF
  "gramp3cstart" = cxAAB6DF
  "gconramp2cend" = cx6497EB
  "gconramp2cstart" = cxFFFFFFFF
  "gramp2cend" = cx548287
  "gramp2cstart" = cxFFFFFFFF
  "gtext" = CX31035E
  "glabel1" = CX31035E
  "gborderlines" = CX31035E
  "goutlines" = CX31035E
  "ggrid" = CX31035E
  "gaxis" = CX31035E
```
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.

```plaintext
style Table from Output /
cellpadding = 5
borderspacing = 2
bordercolordark = colors("tableborderdark")
bordercolorlight = colors("tableborderlight")
borderswidth = 2;
```

Specify attributes for the GraphLabelText element. This STYLE statement is applied to the graph's label text. A DROP_SHADOW attribute is applied.

```plaintext
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.

```plaintext
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.

```plaintext
"Graph axis color attributes" /
linecolordark = colors("tableborderdark")
linecolorlight = colors("tableborderlight")
width = 2;
```
Specify attributes for the `GraphAxisLines` 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.

```plaintext
style GraphAxisLines from GraphAxisLines
   "Axis line attributes" /
      width = 2;
```

Specify attributes for the `GraphBorderLines` element. This `STYLE` statement is applied to the graph's border lines. The width is 2 and the foreground color defined in `Gaxis`, which is CX31035E, is used.

```plaintext
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%.

```plaintext
style GraphCharts from GraphCharts
   "Chart Attributes" /
      transparency = 0.25;
```

Specify attributes for the `GraphWalls` element. This `STYLE` statement is applied to the walls inside the graph's axes. The `GRADIENTDIRECTION` is set to Xaxis, meaning that the gradient is going left to right. The `ENDCOLOR` (CXFFFFFF) is defined in `Gwalls` and is the final color used with the gradient. The data elements of the graph have a `TRANSPARENCY` of 100%. Because a `STARTCOLOR` is not specified, the beginning of the gradient is completely transparent.

```plaintext
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.

```plaintext
end;
run;
```

---

Example 5: Defining Multiple Style Elements in One `STYLE` Statement

**Features:**

- `DEFINE STYLE` statement
  - `FROM` option
  - `Style attribute`: `BACKGROUNDCOLOR`
  - `Style attribute`: `BORDERWIDTH`
  - `Style attribute`: `BORDERSPACING`
  - `Style attribute`: `FONTFAMILY`
  - `Style attribute`: `FONTSTYLE`
  - `Style attribute`: `FONTWEIGHT`
  - `Style attribute`: `COLOR`

- `CLASS` statement

- Other ODS features
  - `ODS HTML` statement
  - `FILE` statement with `ODS=` option
PUT statement with _ODS_ argument

Data set: Grain_Production
Format: $CNTRY.

Details

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

```{r}
filename odsout ".";
ods html close;
opts nodate;
proc template;
define style Styles.NewStyle;
  style cellcontents, header, systemtitle /
    fontfamily="arial, helvetica"
    fontweight=medium
    backgroundcolor=blue
    fontstyle=roman
    fontsize=5
    color=white;
  class header /
    backgroundcolor=very light blue;
  class systemtitle /
    backgroundcolor=white
    color=red
    fontstyle=italic
    fontsize=6;
  style footer from systemtitle /
    fontsize=3;
  class table /
    borderspacing=5
    borderwidth=10;
end;
run;
ods html path=odsout file="newstyle-body.htm" style=newstyle;
title "Leading Grain Producers";
title2 "in 1996";
```
data _null_;  
set grain_production;  
where type  in ("Rice", "Corn") and year=1996;  
file print ods=(
    template="table1"
    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"))
    )
);  
put _ods_;  
run;
ods html close;
ods html;
proc template;  
delete Styles.NewStyle;
run;

Program Description

Create a file reference for the output and set the SAS system options. The current working directory is specified in this example.

filename odsout ".";
ods html close;
opts nodate;


proc template;
   define style Styles.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 in 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 style 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.

```plaintext
class header /
  backgroundcolor=very light blue;
```

**Modify the SystemTitle style element.** By default, ODS uses SystemTitle to produce SAS titles.

```plaintext
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.

```plaintext
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.

```plaintext
class table /
  borderspacing=5
  borderwidth=10;
end;
run;
```

**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 Styles.NewStyle as the style when it formats the output.

```plaintext
ods html path=odsout file="newstyle-body.htm"
  style=newstyle;
```

**Specify the titles for the report.** The TITLE statements provide two titles for the output.

```plaintext
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.

```plaintext
data _null_; 
  set grain_production;
  where type in ("Rice", "Corn") and year=1996;
run;
```

**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. The TEMPLATE= suboption tells ODS to use the table template named Table1, which was previously created with PROC TEMPLATE.

For more information about using the DATA step with ODS, see “Using ODS with the DATA Step” in SAS Output Delivery System: User’s Guide. For the program that
creates the table template Table1, see “Creating the Table1 Table Template” in SAS Output Delivery System: User’s Guide.

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;

Close the HTML destination. The ODS HTML statement closes the HTML destination and all the files that are associated with it. Specify the ODS HTML statement again to return ODS to its default setup.

    ods html close;
    ods html;

Once you have created your output, you can delete the custom styles.

    proc template;
    delete Styles.NewStyle;
    run;

HTML Output: Specifying Colors and Fonts

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 TableFooter 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.
Example 6: Importing a CSS File

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

Details

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 enables 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.

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. Copy and paste this code into a text editor and save it as StyleSheet.css.

```css
@import url(print.css);

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;
}

@media screen {
  .header, .rowheader, .footer, .rowfooter {
    color: white;
    background-color: green;
  }
  .table {
    background-color: yellow;
    border-spacing: 0;
    border: 1px black solid;
  }
  .proctitle {
    font-family: helvetica, sans-serif;
    font-size: x-large;
    font-weight: normal;
  }
}

@media print {
  .header, .rowheader, .footer, .rowfooter, {"color": "white;"}
  .table {
    background-color: blue;
    padding: 5px;
  }
  .data {
    font-size: small;
  }
}
Program

filename odsout ".";
ods html close;
options nodate;
proc template;
define style styles.mycssstyle;
   import "StyleSheet.css";
   class data /
      color = red;
end;

define style styles.mycssstyleprinter;
   parent=styles.mycssstyle;
   import "StyleSheet.css" print;
end;
run;

ods html path=odsout file="css.html" style=styles.mycssstyle;
ods pdf file="your-file-path/css.pdf" style=styles.mycssstyleprinter;

proc contents data=sashelp.class;
run;
ods _all_ close;
ods html;

Program Description

Create a file reference for the output and set the SAS system options. The current working directory is specified in this example.

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.

Specifying class data / color=red; is the same as specifying style data from data / color=red;.

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.

```sas
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 filename of the output, and the style to use. The CONTENTS procedure shows the contents of the SAS data set Sashelp.Class.

```sas
ods html path=odsout file="css.html" style=styles.mycssstyle;
ods pdf file="your-file-path/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. Specify the ODS HTML statement to return ODS to its default setup.

```sas
ods _all_ close;
ods html;
```
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

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>KANHELP.GCLASS</th>
<th>Observations</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>5</td>
</tr>
<tr>
<td>Engine</td>
<td>V3</td>
<td>Indexes</td>
<td>9</td>
</tr>
<tr>
<td>Created</td>
<td>03/10/2011 10:14:32</td>
<td>Observation Length</td>
<td>40</td>
</tr>
<tr>
<td>Last Modified</td>
<td>03/10/2011 10:14:32</td>
<td>Deleted Observations</td>
<td>9</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
<td>Compressed</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td></td>
<td>Sorted</td>
<td>NO</td>
</tr>
<tr>
<td>Label</td>
<td>Student Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td></td>
<td>WINDOWS_32</td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>YASASCII 8 ASCII (ANR)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Engine/Host Dependent Information

<table>
<thead>
<tr>
<th>Data Set Page Size</th>
<th>53734</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Data Set Pages</td>
<td>1</td>
</tr>
<tr>
<td>First Data Page</td>
<td>1</td>
</tr>
<tr>
<td>Max Obs per Page</td>
<td>16382</td>
</tr>
<tr>
<td>Obs in First Data Page</td>
<td>10</td>
</tr>
<tr>
<td>Number of Data Set Records</td>
<td>0</td>
</tr>
<tr>
<td>ExtendObsCounter</td>
<td>YES</td>
</tr>
<tr>
<td>Filename</td>
<td>C:/SAS/9.2/sample/dev/ora/v3/put/FILL.DAT</td>
</tr>
<tr>
<td>Release Created</td>
<td>5.001B0</td>
</tr>
<tr>
<td>Host Created</td>
<td>NET_8RV</td>
</tr>
</tbody>
</table>

#### Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Age</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Height</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>Name</td>
<td>Char</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Sex</td>
<td>Char</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Weight</td>
<td>Num</td>
<td>8</td>
</tr>
</tbody>
</table>
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.
Example 7: Table Header and Footer Border Formatting

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 statement
ODS SELECT statement

Data set: Stats and Stats2

Details
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 valid only in the RTF destination.

Program
ods html close;
options nodate;
title "TableHeaderContainer, TableFooterContainer, and Border Control Style Attributes";
title2 "Allows Control of Borders Between the Header, Body, and Footer of a Table";
proc template;
define style HeadersFootersBorders;
parent=styles.rtf;
style TableHeaderContainer from TableHeaderContainer /
bordertopwidth=6
bordertopcolor=red
bordertopstyle=double;
styple TableFooterContainer from TableFooterContainer /
bordertopwidth=6
bordertopcolor=red
bordertopstyle=double;
dstyle table from table /
borderspacing=0 rules=groups frame=void;
end;
run;
proc template;
edit Base.Datasets.Members;
    header hd1;
    footer ft1;
define hd1;
preformatted=on;
just=1;
text" Table Header with Leading and Trailing Blanks ";
end;
define ft1;
preformatted=on;
just=1;
text" Table Footer with Leading and Trailing Blanks ";
end;
Program Description

Close the HTML destination so that no HTML output is produced. The HTML destination is open by default. The ODS HTML CLOSE statement closes the HTML destination to conserve resources. If the destination were left open, then ODS would produce both HTML and PDF output.

ods html close;
options nodate;

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

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 /
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 BORDERSPACING=, RULES=, and FRAME= attributes modify the border spacing, rules, and frame of the table.

```
style table from table /
    borderspacing=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.

```
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.

```plaintext
ods rtf file="your-file-path/headerfooters.rtf" style=HeadersFootersBorders;
ods select members;
proc datasets lib=work;
run;
quit;
```

Close the RTF destination and open the HTML destination. The ODS RTF CLOSE statement closes the RTF 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. Specify the ODS HTML statement to return ODS to its default setup.

```plaintext
ods rtf close;
ods html;
```

Once you have created your output, you can delete the custom styles.

```plaintext
proc template;
delete HeadersFootersBorders;
delete Base.Datasets.Members;
run;
```

**RTF Output**

*(Output 14.11)* RTF Output with Custom Headers and Footers

---

Example 8: Enhancing Titles and Footnotes in PDF Output

**Features:**
- CLASS statement features
  - COLOR= style attribute
  - FONT_WEIGHT= style attribute
- SystemFooter style element
Details

The text of titles and footnotes created by tabular (non-graphical) output is controlled using the user-defined style attribute TitleFont in the style element Fonts. In Styles.Pearl (the default style for PDF output), the source code shows that for TitleFont, the font family is "<MTsans-serif>, Albany", the font size is 11pt, and the font weight is bold:

source styles.pearl;
define style Styles.Pearl;
parent = styles.printer;
class fonts /
   'TitleFont2' = ("<MTsans-serif>, Albany",10pt,bold)
   'TitleFont' = ("<MTsans-serif>, Albany",11pt,bold)
   'StrongFont' = ("<MTsans-serif>, Albany",8pt,bold)
   'EmphasisFont' = ("<MTsans-serif>, Albany",8pt,italic)
   'FixedEmphasisFont' = ("<MTmonospace>, Courier",7pt)
   'FixedStrongFont' = ("<MTmonospace>, Courier",7pt,bold)
   'FixedHeadingFont' = ("<MTmonospace>, Courier",7pt,bold)
   'BatchFixedFont' = ("SAS Monospace, <MTmonospace>, Courier",6pt)
   'FixedFont' = ("<MTmonospace>, Courier",7pt)
   'headingEmphasisFont' = ("<MTsans-serif>, Albany",9pt,bold italic)
   'headingFont' = ("<MTsans-serif>, Albany",8pt,bold)
   'docFont' = ("<MTsans-serif>, Albany",8pt);

The following example uses PROC TEMPLATE to edit the style attribute TitleFont to make the following changes:

- font size 14pt
- font weight medium
- font style italic

For separate control of the footnotes, the style element SystemFooter can be modified using PROC TEMPLATE to make the footnotes green and medium weight.

Program

ods html close;
proc template;
   define style styles.MyPDFstyle;
   parent=styles.pearl;
   class fonts /
      'TitleFont2' = ("<MTsans-serif>, Albany",10pt,bold)
      'TitleFont' = ("<MTsans-serif>, Albany",14pt, medium italic)
      'StrongFont' = ("<MTsans-serif>, Albany",8pt,bold)
      'EmphasisFont' = ("<MTsans-serif>, Albany",8pt,italic)
      'FixedEmphasisFont' = ("<MTmonospace>, Courier",7pt)
      'FixedStrongFont' = ("<MTmonospace>, Courier",7pt,bold)
      'FixedHeadingFont' = ("<MTmonospace>, Courier",7pt,bold)
      'BatchFixedFont' = ("SAS Monospace, <MTmonospace>, Courier",6pt)
      'FixedFont' = ("<MTmonospace>, Courier",7pt)
      'headingEmphasisFont' = ("<MTsans-serif>, Albany",9pt,bold italic)
      'headingFont' = ("<MTsans-serif>, Albany",8pt,bold)
      'docFont' = ("<MTsans-serif>, Albany",8pt);
Program Description

Close the HTML destination so that no HTML output is produced. The HTML destination is open by default. The ODS HTML CLOSE statement closes the HTML destination to conserve resources. If the destination were left open, then ODS would produce both HTML and PDF output.

ods html close;

Create a new style named Styles.MyPDFstyle that inherits from Styles.Pearl. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style called Styles.MyPDFstyle. The PARENT= attribute specifies Styles.Pearl as the style from which Styles.MyPDFstyle inherits. All the style elements, attributes, and statements that are specified in the parent’s style template (Pearl) are used in the child style template (MyPDFstyle) unless the child style template overrides them.

proc template;
    define style styles.MyPDFstyle;
    parent=styles.pearl;
end;

Customize the title font by changing the attributes of the style element Fonts. The style element Fonts contains the style attribute that controls the appearance of titles. This CLASS statement adds the style element Fonts to the custom style Styles.MyPDFstyle. The Fonts style element also exists in the parent style (Styles.Pearl). The CLASS statement also changes the values of the user-defined
Example 8: Enhancing Titles and Footnotes in PDF Output

style attribute TitleFont2. The change is that the title font will now be 14pt, medium weight, and italic.

class fonts /
   'TitleFont2' = ('<MTsans-serif>, Albany",14pt,bold)
   'TitleFont' = ('<MTsans-serif>, Albany",14pt, medium italic)
   'StrongFont' = ('<MTsans-serif>, Albany",8pt,bold)
   'EmphasisFont' = ('<MTsans-serif>, Albany",8pt,italic)
   'FixedEmphasisFont' = ('<MTmonospace>, Courier",7pt)
   'FixedStrongFont' = ('<MTmonospace>, Courier",7pt,bold)
   'FixedHeadingFont' = ('<MTmonospace>, Courier",7pt,bold)
   'BatchFixedFont' = ('SAS Monospace, <MTmonospace>, Courier",6pt)
   'FixedFont' = ('<MTmonospace>, Courier",7pt)
   'headingEmphasisFont' = ('<MTsans-serif>, Albany",9pt,bold italic)
   'headingFont' = ('<MTsans-serif>, Albany",8pt,bold)
   'docFont' = ('<MTsans-serif>, Albany",8pt);
end;

Customize the footer by adding the SystemFooter style element. The SystemFooter style element controls the appearance of footers. It is not a default style element that is present in the parent style (Styles.Pearl). The CLASS statement adds SystemFooter, which contains the style attributes COLOR= and FONT_WEIGHT=, to Styles.MyPDFstyle. This statement results in the font of the footnote being changed to a medium, green font.

class SystemFooter /
   color = green
   font_weight=medium;
end;
run;

Set the SAS system options.

options nodate number;

Create the PDF file and specify the custom style. The ODS PDF statement creates the PDF file. The STYLE= option specifies that the custom style template Styles.MyPDFstyle is applied to the PDF output.

ods pdf file="file.pdf" style=styles.MyPDFstyle;

Create the PRINT procedure output and specify a title and footnote. The appearance of the title and footnote reflects the changes made in the style MyPDFstyle.

proc print data=sashelp.cars(obs=5);
   title "First 5 observations from SASHELP.CARS";
   footnote "Created by SAS &sysver";
run;

Close the PDF destination and open the HTML destination. The ODS PDF CLOSE statement closes 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. The ODS HTML statement opens the HTML destination and returns ODS to its default setting.

ods pdf close;
ods html;

Remove the customized style template. The DELETE statement removes the customized style that was created in this example. When you use the DELETE
statement, ODS looks for the Styles.MyPDF style in Sasuser.Templat first. If it is there, it deletes it. If not, it searches Sashelp.Tmplmst.

```
proc template;
delete Styles.MyPDFstyle;
end;
title;
```

PDF Output

**Output 14.12** PDF Output with Custom Title and Footnote

![PDF Output Example](image)

**Example 9: Customizing Graphic and Tabular Titles in PDF Output**

Features:

- STYLE statement features
  - COLOR= style attribute
  - FROM option
  - SystemTitle style element
  - SystemTitle2 style element
  - SystemTitle3 style element
  - TitlesandFooters style element
- DEFINE STYLE statement
- PARENT= statement
- OPTIONS statement
- ODS PDF statement options
  - NOGTITLE
  - NOTOC
- PROC PRINT
- PROC SGPANEL
- TITLE statement
Details

The style elements and attributes that control the appearance of titles in tabular output are in the Base.Template.Style template. In the BASE.TEMPLATE.STYLE definition, there is an inheritance defined for each title (and footnote) as shown by this code snippet:

```plaintext
style SystemTitle from TitlesAndFooters
  "Controls system title text."
style SystemTitle2 from SystemTitle
  "Controls system title2 text"
style SystemTitle3 from SystemTitle2
  "Controls system title3 text"
```

When you use a style template to customize titles in PDF output, you must be careful if you have graphical output in your PDF file. By default, titles and footnotes are embedded in graphical output created by ODS graphics and SAS/Graph. The style definitions contain separate definitions for the titles and footnotes for these procedures.

The fonts controlling graphical output are defined under the Graphfonts collection in the default PDF style Styles.Pearl, as shown in this code snippet:

```plaintext
..more style information...
class GraphFonts /
  'GraphAnnoFont' = ('<MTsans-serif>, Albany",10pt)
  'GraphTitle1Font' = ('<MTsans-serif>, Albany",14pt,bold)
  'GraphTitleFont' = ('<MTsans-serif>, Albany",11pt,bold)
  'GraphFootnoteFont' = ('<MTsans-serif>, Albany",10pt)
  'GraphLabelFont' = ('<MTsans-serif>, Albany",10pt)
  'GraphLabel2Font' = ('<MTsans-serif>, Albany",9pt)
  'GraphValueFont' = ('<MTsans-serif>, Albany",9pt)
  'GraphUnicodeFont' = ('<MTsans-serif-unicode">,9pt)
  'GraphDataFont' = ('<MTsans-serif>, Albany",7pt);
..more style information...
```

This example illustrates the following points

- You can use the ODS PDF option NOGTITLE to specify that the titles for any graphical output are not embedded in the resulting image. This enables the titles to be under the control of Systemtitle and Systemtitle, which are the style elements that control titles in tabular output. This is also true for footnotes in PDF output.

- Most SAS styles are written with a large degree of inheritance. There are very few stand-alone styles. After all of your output titles are controlled by Systemtitle and Systemtitle, you can use inheritance to create a custom style that customizes the titles for your PDF output.

Program

```plaintext
ods html close;
proc template;
  define style Styles.CustomTitles;
  parent=styles.pearl;
  style SystemTitle from TitlesAndFooters
    "Controls system title text." / color=blue;
```
style SystemTitle2 from SystemTitle
   "Controls system title2 text" / color=orange;
style SystemTitle3 from SystemTitle2
   "Controls system title3 text" / color=blue;
end;
run;
ods pdf file="PDFStyle.pdf" notoc style=Styles.CustomTitles nogtitle;
title "Predicted and Actual Sales";
title2 "For Sofas";
title3 "By Region";
options nodate nonumber obs=50000;
proc sort data=sashelp.prdsale out=prdsale;
   by Country;
run;
proc sgpanel data=prdsale;
   where quarter=1;
   panelby product / novarname;
   vbar region / response=predict;
   vline region / response=actual lineattrs=GraphFit;
   colaxis fitpolicy=thin;
   rowaxis label='Sales';
run;
options nodate nonumber obs=8;
proc print data=prdsale;
   var product region actual predict;
run;
ods pdf close;
ods html;

Program Description

Close the HTML destination so that no HTML output is produced. The HTML
destination is open by default. The ODS HTML CLOSE statement closes the HTML
destination to conserve resources. If the destination were left open, then ODS would
produce both HTML and PDF output.

ods html close;

Create a custom style template that modifies the appearance of titles. This
PROC TEMPLATE step creates a new style named Styles.CustomTitles that inherits
all of its style elements and attributes from Styles.Pearl. The STYLE statements use
the COLOR= style attribute to change the color of the system options in tabular
output.

proc template;
   define style Styles.CustomTitles;
   parent=styles.pearl;
   style SystemTitle from TitlesAndFooters
      "Controls system title text." / color=blue;
   style SystemTitle2 from SystemTitle
      "Controls system title2 text" / color=orange;
   style SystemTitle3 from SystemTitle2
      "Controls system title3 text" / color=purple;
end;
Open the PDF destination and specify the ODS PDF statement options. The NOTOC option specifies that no table of contents is created. The NOGTITLE option specifies that titles are not embedded in graphical output. This enables the titles to be controlled by the style elements specified in the previous STYLE statements.

```
ods pdf file="PDFStyle.pdf" notoc style=Styles.CustomTitles nogtitle;
```

Specify the titles.

```
title "Predicted and Actual Sales";
title2 "For Sofas"
title3 "By Region"
```

Create the procedure output.

```
options nodate nonumber obs=50000;
proc sort data=sashelp.prdsale out=prdsale;
  by Country;
run;

proc sgpanel data=prdsale;
  where quarter=1;
  panelby product / novarname;
  vbar region / response=predict;
  vline region / response=actual lineattrs=GraphFit;
  colaxis fitpolicy=thin;
  rowaxis label='Sales';
run;
```

Close the PDF destination and open the HTML destination. The ODS PDF CLOSE statement closes 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. The ODS HTML statement opens the HTML destination and returns ODS to its default setting.

```
ods pdf close;
ods html;
```

PDF Output

The following output shows that the custom style is applied to all of our output.
If you omit the NOGTITLE option from the ODS PDF statement, then the titles are embedded within the graph and the custom style is not applied to them.
Example 9: Customizing Graphic and Tabular Titles in PDF Output

Output 14.14  Custom Style not Applied to Graphical Output

![Graphical Output]

Predicted and Actual Sales
For Sofas
By Region

<table>
<thead>
<tr>
<th>Region</th>
<th>BED</th>
<th>CHAIR</th>
<th>DESK</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sales

$0
$5,000
$10,000
$15,000
$20,000

EAST     WEST     EAST     WEST     EAST     WEST

Predicted Sales  Actual Sales
TEMPLATE Procedure: Creating Tabular Templates

Overview: TEMPLATE Procedure: Creating Tabular Templates ... 515
Using the TEMPLATE Procedure to Create or Customize Tabular Output ... 516
What You Can Do with Table Templates ... 517

Concepts: TEMPLATE Procedure: Creating Tabular Templates ... 520
Comparing the Edit of an Existing Table Template with Creating a New Table Template ... 520
Viewing the Contents of a Table Template ... 520

Syntax: TEMPLATE Procedure: Creating Tabular Templates ... 521
CELLSTYLE AS Statement ... 523
COLUMN Statement ... 526
COMPUTE AS Statement ... 527
DEFINE Statement ... 529
DEFINE COLUMN Statement ... 530
DEFINE FOOTER Statement ... 532
DEFINE HEADER Statement ... 532
DEFINE TABLE Statement ... 534
DYNAMIC Statement ... 535
EDIT Statement ... 536
END Statement ... 537
FOOTER Statement ... 537
HEADER Statement ... 538
MVAR Statement ... 539
NMVAR Statement ... 540
NOTES Statement ... 541
TEXT Statement ... 541
TEXT2 Statement ... 542
TEXT3 Statement ... 543
TRANSLATE INTO Statement ... 544

Usage: TEMPLATE Procedure: Creating Tabular Templates ... 548
Values in Table Columns and How They Are Justified ... 548
Formatting Values in Table Columns ... 549
Stacking Values for Two or More Variables ... 550

Examples: TEMPLATE Procedure: Creating Tabular Templates ... 551
Example 1: Editing a Table Template That a SAS Procedure Uses ... 551
Example 2: Comparing the EDIT Statement to the DEFINE TABLE Statement ... 557
Example 3: Creating a New Table Template ... 563
Example 4: Setting the Style Element for Cells Based on Their Values ... 571
Example 5: Setting the Style Element for a Specific Column, Row, and Cell ... 576
Example 6: Creating Master Templates ... 583
Example 7: Table Header and Footer Border Formatting ... 586
Overview: TEMPLATE Procedure: Creating Tabular Templates

Using the TEMPLATE Procedure to Create or Customize Tabular Output

The TEMPLATE procedure enables you to customize the tabular appearance of your SAS output.

Tabular templates describe how tables should be constructed. This includes the content and placement of headers and footers, the content and placement of columns, and style overrides. All SAS procedures, except PROC PRINT, PROC REPORT, and PROC TABULATE, use tabular templates to describe how their tabular output should look. This means that you can change the structure of tables that are generated by SAS procedures by using tabular templates.

With the TEMPLATE procedure, you can create and modify tabular templates. Tabular templates include the following types:

- column templates
- header templates
- footer templates
- table 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 your 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 15.1  PROC TEMPLATE Statements

<table>
<thead>
<tr>
<th>Customization</th>
<th>Element Modified</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column presentation</td>
<td>Column template</td>
<td>&quot;DEFINE COLUMN Statement&quot; on page 530</td>
</tr>
<tr>
<td>Table footer</td>
<td>Footer template</td>
<td>&quot;DEFINE FOOTER Statement&quot; on page 532</td>
</tr>
<tr>
<td>Table header</td>
<td>Header template</td>
<td>&quot;DEFINE HEADER Statement&quot; on page 532</td>
</tr>
</tbody>
</table>
Customization | Element Modified | Statement
--- | --- | ---
Single output object | Table template | “DEFINE TABLE Statement” on page 534
An existing template for a table, column, header, or footer | Table, column, header, footer | “EDIT Statement” on page 536

You can find additional tips and tricks in the SAS press book excerpt, ODS Techniques: Tips for Enhancing Your SAS Output, by Kevin Smith. This SAS press book is a cookbook-style collection of Kevin’s top ODS tips and techniques to teach you how to bring your reports to a new level and inspire you to see ODS in a new light.

What You Can Do with Table Templates

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.

Figure 15.1 LISTING Output from PROC UNIVARIATE (Default Moments Table)
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.
**Figure 15.3**  LISTING Output from PROC UNIVARIATE (Customized Moments Table)

**Figure 15.4**  RTF Output of Sales Statistics from PROC UNIVARIATE (Customized Moments Table)
Concepts: TEMPLATE Procedure: Creating Tabular Templates

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: Assuming the default ODS path, 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” in SAS Output Delivery System: User’s Guide for more information.

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 want 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

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.

```sas
proc template;
source base;
run;
```

---

### Syntax: TEMPLATE Procedure: Creating Tabular Templates

**PROC TEMPLATE:**

1. **EDIT** `template-path-1 <AS template-path-2> < / STORE=libref.template-store > ; statements-and-attributes END;`  
2. **DEFINE COLUMN** `column-path | Base.Template.Column < / STORE=libref.template-store>; statements-and-attributes END;`  
3. **DEFINE FOOTER** `footer-path | Base.Template.Footer < / STORE=libref.template-store>; statements-and-attributes END;`  
4. **DEFINE HEADER** `template-name | Base.Template.Header; statements-and-attributes END;`  
5. **DEFINE TABLE** `table-path | Base.Template.Table < / STORE=libref.template-store>; statements-and-attributes END;`
<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELLSTYLE AS</td>
<td>Set the style element of the cells in the table or column according</td>
<td>Ex. 5,</td>
</tr>
<tr>
<td></td>
<td>to the values of the variables</td>
<td>Ex. 6</td>
</tr>
<tr>
<td>COLUMN</td>
<td>Declare a symbol as a column in the table and specify the order of</td>
<td>Ex. 2</td>
</tr>
<tr>
<td></td>
<td>the columns</td>
<td></td>
</tr>
<tr>
<td>COMPUTE AS</td>
<td>Compute values for a column that is not in the data component, or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>modify the values of a column that is in the data component</td>
<td></td>
</tr>
<tr>
<td>DEFINE</td>
<td>Create a template inside a table template</td>
<td>Ex. 2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 3,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 7</td>
</tr>
<tr>
<td>DEFINE COLUMN</td>
<td>Create a template for a column</td>
<td>Ex. 4,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 5,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 6</td>
</tr>
<tr>
<td>DEFINE FOOTER</td>
<td>Create a template for a table footer</td>
<td>Ex. 3</td>
</tr>
<tr>
<td>DEFINE HEADER</td>
<td>Create a template for a table header or a header inside a column</td>
<td>Ex. 5,</td>
</tr>
<tr>
<td></td>
<td>template</td>
<td>Ex. 6</td>
</tr>
<tr>
<td>DEFINE TABLE</td>
<td>Create a table template</td>
<td>Ex. 4,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 5,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 6</td>
</tr>
<tr>
<td>DYNAMIC</td>
<td>Define a symbol that references a value that the data component</td>
<td>Ex. 2</td>
</tr>
<tr>
<td></td>
<td>supplies from the procedure or DATA step</td>
<td></td>
</tr>
<tr>
<td>EDIT</td>
<td>Edit an existing template for a table, column, header, or footer</td>
<td>Ex. 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex. 7</td>
</tr>
<tr>
<td>END</td>
<td>End the table template, header template, column template, or footer</td>
<td></td>
</tr>
<tr>
<td>FOOTER</td>
<td>Declare a symbol as a footer in the table and specify the order of</td>
<td>Ex. 7</td>
</tr>
<tr>
<td></td>
<td>the footers</td>
<td></td>
</tr>
<tr>
<td>HEADER</td>
<td>Declare a symbol as a header in the table and specify the order of</td>
<td>Ex. 3,</td>
</tr>
<tr>
<td></td>
<td>the headers</td>
<td>Ex. 7</td>
</tr>
<tr>
<td>MVAR</td>
<td>Define a symbol that references a macro variable, the value of which</td>
<td>Ex. 3</td>
</tr>
<tr>
<td></td>
<td>is treated as a string</td>
<td></td>
</tr>
<tr>
<td>NMVAR</td>
<td>Define a symbol that references a macro variable, the value of which</td>
<td></td>
</tr>
<tr>
<td></td>
<td>is treated as a number</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>Provide information about the table, header, column, or footer</td>
<td>Ex. 2</td>
</tr>
<tr>
<td>TEXT</td>
<td>Specify the text of a header, footer, or the label of a variable in</td>
<td>Ex. 5</td>
</tr>
<tr>
<td></td>
<td>an output data set</td>
<td></td>
</tr>
<tr>
<td>TEXT2</td>
<td>Provide an alternative header or footer to use in the LISTING output</td>
<td></td>
</tr>
<tr>
<td></td>
<td>if the header or footer that is provided by the TEXT statement is</td>
<td></td>
</tr>
<tr>
<td></td>
<td>too long</td>
<td></td>
</tr>
</tbody>
</table>
CELLSTYLE AS Statement

For tables, sets the style element of the cells in the table or column according to the values of the variables. For text, sets the style attributes of the list items or paragraphs. Use this statement to set the presentation characteristics (such as foreground color and font face) of individual cells or text.

Restriction: The CELLSTYLE AS statement can be used only within a column template, an ODS list, an ODS textblock, or a table template.

Syntax

```
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 list item, paragraph, or table cell.

If expression resolves to TRUE (a nonzero 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 string, 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 about SAS functions, see *SAS Functions and CALL Routines: Reference*.

**Example**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXT3</td>
<td>Provide an alternative header or footer to use in the LISTING output if the header or footer that is provided by the TEXT2 statement is too long</td>
</tr>
<tr>
<td>TRANSLATE INTO</td>
<td>Translate the specified numeric values to other values</td>
</tr>
</tbody>
</table>
**built-in variable**

is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

**_COLUMN_**

is a column number. Column numbering begins with 1.

**Alias**

_COL_

**Example**

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

**_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:

```plaintext
cellstyle _datatype_ = "num" as {color=red},
    _datatype_ = "char" as {color=blue};
```

**_LABEL_**

is a column label.

**Example**

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

**_ROW_**

is a row number. Row numbering begins with 1.

**Example**

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

**_STYLE_**

is a style element name.

**See**

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**Example**

“Example 6: Creating Master Templates” on page 583

**_VAL_**

is the data value of a cell.

**Tip**

Use _VAL_ to represent the value of the current column.

**Example**

“Example 6: Creating Master Templates” on page 583

---

**comparison-operator**

compares a variable with a value or with another variable.

The following table lists the comparison operators:
### Table 15.2 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>

**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. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**See**
You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

*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 about the style attributes that you can set in a table template, see “About Style Attributes” on page 459.

**Optional Argument**

*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 by using the “DEFINE STYLE Statement” on page 448. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.
The following style elements are most likely to be used with the CELLSTYLE AS statement:

- Data
- DataFixed
- DataEmpty
- DataEmphasis
- DataEmphasisFixed
- DataStrong
- DataStrongFixed
- ListItem
- ListItem2
- Paragraph

The style element provides the basis for displaying the cell. Additional style attributes modify the display.

Default: Data

See Chapter 14, "TEMPLATE Procedure," on page 425

For a table of style element names, see Chapter 20, "Style Elements," on page 797.

COLUMN Statement

Declares a symbol as a column in the table and specifies the order of the columns.

Restriction: The COLUMN statement can be used only within a table template.

Examples:

"Example 3: Creating a New Table Template" on page 563
"Example 2: Defining Variables with the COLUMN Statement" on page 270

Syntax

```
COLUMN column(s);
```

Required Argument

`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.

Defaults

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.

**See**

“Stacking Values for Two or More Variables ” on page 550

---

**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.

**Restriction:** The COMPUTE AS statement can be used only within a column template.

**Syntax**

**COMPUTE AS** expression;

**Required Argument**

**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:

```plaintext
define column sqroot;
    compute as sqrt(source);
    header="Square Root";
    format=6.4;
```
end;

function

specifies a SAS function. For information about SAS functions, see SAS Functions and CALL Routines: Reference.

built-in variable

is a special type 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_

Example  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_DATANAME_

is a data-column name.

_LABEL_

is a column label.

Example  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_ROW_

is a row number. Row numbering begins with 1.

Example  “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_STYLE_

is a style-element name.

Example  “Example 6: Creating Master Templates” on page 583

_VAL_

is the data value of a cell.

Tip  Use _VAL_ to represent the value of the current column.

Example  “Example 6: Creating Master Templates” on page 583

**comparison-operator**

compares a variable with a value or another variable.

**Table 15.3  Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>Symbol</td>
<td>Mnemonic Equivalent</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

**Tip**  
The COMPUTE AS statement can alter values in an output object. None of the templates that SAS provides modifies any values. To determine whether a template was provided by SAS, use the “ODS VERIFY Statement” in SAS Output Delivery System: User’s Guide. 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 whether the COMPUTE AS statement alters values. (See “SOURCE Statement” on page 346.)

**See**  
You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

**Example**  
“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

### DEFINE Statement

Creates a template inside a table template.

**Restriction:**  
The DEFINE statement can be used only inside a table template.

**See:**  
“DEFINE COLUMN Statement” on page 530  
“DEFINE FOOTER Statement” on page 532  
“DEFINE HEADER Statement” on page 532

**Example:**  
“Example 2: Defining Variables with the COLUMN Statement” on page 270

**Syntax**

```
DEFINE <COLUMN | FOOTER | HEADER> template-name </ options>
    statements-and-attributes;
END;
```
**Required Argument**

*template-name*

specifies the name of the new object.

**Restriction**

*template-name* must be a single-level name.

**Tip**

To reference the template that you are creating from another template, create it outside the table template.

---

**Optional Arguments**

**COLUMN | FOOTER | HEADER**

specifies the type of template to create.

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-type* is optional if you specify the COLUMN name before the definition.

The same is true for headers and footers.

**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.

---

**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:**

“DEFINE HEADER Statement” on page 532

**Examples:**

“Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

“Example 6: Creating Master Templates” on page 583

---

**Syntax**

DEFINE COLUMN column-path | Base.Template.Column

< / STORE=libref.template-store>

<column-attribute-1 < column-attribute-2>…>;

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;
DEFINE COLUMN Statement

```
statements-and-attributes
END;
```

**DYNAMIC** `variable-1 <=default-variable-1`
```
<text-1>
<... variable-n <=default-variable-n><text-n>>;
```

**MVAR** `variable-1 <=default-variable-1><text-1>
<... variable-n <=default-variable-n><text-n>>;
```

**NMVAR** `variable-1 <=default-variable-1><text-1>
<... variable-n <=default-variable-n><text-n>>;
```

**NOTES** "text";

**TRANSLATE** `expression-1 INTO expression-2
<, expression-n INTO expression-m>;
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.

**Restrictions**

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.

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 tabular templates.

**Example**

“Example 6: Creating Master Templates” on page 583

### Optional Argument

**STORE=libref.template-store**

Specifies the template store in which to store the template. If the template store does not exist, it is created.

**Restrictions**

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.

The `STORE=` option does not become part of the template.
DEFINE FOOTER Statement

Creates a template for a table footer.

Requirement: An END statement must be the last statement in the template.

See: “DEFINE HEADER Statement” on page 532

Examples: “Example 3: Creating a New Table Template ” on page 563
“Example 1: Creating a Stand-Alone Style” on page 459

Syntax

DEF**INE FOOTER** footer-path | Base.Template.Footer

< / STORE=libref.template-store>;  
<header/footer-attribute-1 < header/footer-attribute-2> ...>;  
**DYNAMIC** variable-1 <=default-variable-1>  
`<text-1>`  
... variable-n <=default-variable-n>`text-n'>;  
**MVAR** variable-1 <=default-variable-1>`text-1'>  
... variable-n <=default-variable-n>`text-n'>;  
**NMVAR** variable-1 <=default-variable-1>`text-1'>  
... variable-n <=default-variable-n>`text-n'>;  
NOTES "text";  
**TEXT** footer-specification;  
**TEXT2** footer-specification;  
**TEXT3** footer-specification;  
END;

Substatements and Attributes

The substatements in the DEFINE FOOTER statements and the footer attributes are the same as the substatements in the “DEFINE HEADER Statement” on page 532 and the “Header and Footer Attributes” on page 608.

DEFINE HEADER Statement

Creates a template for a table header or a header inside a column template.

Restriction: The DEFINE HEADER statement can be used only within a column template or a table template.

Requirement: An END statement must be the last statement in the template.

See: “Example 3: Creating a New Table Template ” on page 563
Syntax

**DEFINE HEADER**  
`header-path | Base.Template.Header`  
```  
< / STORE=libref.template-store > ;  
< header/footer-attribute-1 < header/footer-attribute-2 > ... > ;  
**DYNAMIC** `variable-1 <= default-variable-1`  
`< 'text-1' >`  
`< ... variable-n <= default-variable-n > < 'text-n' > > ;`  
**MVAR** `variable-1 <= default-variable-1 > < 'text-1' >`  
`< ... variable-n <= default-variable-n > < 'text-n' > > ;`  
**NMVAR** `variable-1 <= default-variable-1 > < 'text-1' >`  
`< ... variable-n <= default-variable-n > < 'text-n' > > ;`  
**NOTES** "text" ;  
**TEXT** `header-specification ;`  
**TEXT2** `header-specification ;`  
**TEXT3** `header-specification ;`  
**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.

Restrictions

If the template is nested inside another template, `header-path` must be a single-level name.

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. After 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 tabular templates.

Example

“Example 6: Creating Master Templates” on page 583

Optional Argument

**STORE=libref.template-store**

Specifies the template store in which to store the template. If the template store does not exist, it is created.
Restrictions

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.

The STORE= option does not become part of the template.

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.

Examples:

“Example 3: Creating a New Table Template ” on page 563
“Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

Syntax

```
DEFINE TABLE table-path | Base.Template.Table
  <$ STORE=libref.template-store >;
  <table-attribute-1 < table-attribute-2>...>;
  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 <$ options >;
    statements-and-attributes
  END;
  DYNAMIC variable-1 <=default-variable-1>
    <$ text-1$><… variable-n <=default-variable-n><text-n$>;
  FOOTER footer-name(s);
  HEADER header-name(s);
  MVAR variable-1 <=default-variable-1>$text-1$>
    <$… variable-n <=default-variable-n><text-n$>
  NMVAR variable-1 <=default-variable-1>$text-1$>
    <$… variable-n <=default-variable-n><text-n$>
  NOTES "$text$;
  TRANSLATE expression-1 INTO expression-2 <$ , expression-n INTO expression-m >;
  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.

<table>
<thead>
<tr>
<th>Interaction</th>
<th>The Base.Template.Table master template attributes are overridden by other tabular templates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>The Base.Template.Table master template is most useful when used with the CELLSTYLE AS statements to create alternating colors in your tabular output.</td>
</tr>
<tr>
<td>Example</td>
<td>“Example 6: Creating Master Templates” on page 583</td>
</tr>
</tbody>
</table>

**Optional Argument**

**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. |

---

**DYNAMIC Statement**

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

**Restriction:**

The DYNAMIC statement can be used only in the template of an ODS textblock, ODS list, 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.

**Syntax**

```
DYNAMIC `variable-name-1` <=`value-1`<`text-1`>>
   < `variable-name-2` <=`value-2`<`text-2`...>>;
```

**Required Argument**

`variable-name`

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. |
Optional Arguments

*value*

sets the value of the variable.

*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.

## EDIT Statement

Edits an existing template. The EDIT statement replaces the DEFINE statement in a template block when editing. You can use the EDIT statement in place of any DEFINE statement.

**Restriction:**
If you edit a template that is a link, the link is broken and a separate template is created.

**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.

**Example:**
“Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551

### Syntax

```
EDIT template-path-1 <AS template-path-2 > </ STORE=libref.template-store>;
  template-statements;
END;
```

**Required Argument**

*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 76.)

### Optional Arguments

**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.
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`.

**template-statements**

`template-statements` are any statements or attributes that are valid between the DEFINE statement and the END statement.

**Editing an Existing Template**

When you use the EDIT statement, the following occurs:

- 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 344.) It opens a copy of the first template path that it finds in a template store that has Read access.

- 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.

**END Statement**

Ends the table template, header template, column template, or footer template.

**Syntax**

```r
END;
```

**FOOTER Statement**

Declares a symbol as a footer in the table and specifies the order of the footers.

**Example:**

“Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390
Syntax

**FOOTER** *footer-specification(s)*;

Required Argument

*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 and PROC ODS TABLE treat it as the split character.

See "SPLIT= "character" | variable;" on page 617

*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 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.

---

**HEADER Statement**

Declares a symbol as a header in the table and specifies the order of the headers.

Example: "Example 1: Creating a Customized Crosstabulation Table Template with No Legend" on page 390

Syntax

**HEADER** *header-specification(s)*;

Required Argument

*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 and PROC ODSTABLE treat it as the split character.

See “SPLIT= "character" | variable;” on page 617

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.

Example “Example 3: Creating a New Table Template ” on page 563

---

MVAR Statement

Defines a symbol that references a macro variable. ODS uses 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.

Restriction: When replaying an ODS document with PROC DOCUMENT, values created by the MVAR statement must be re-created in the same session that is replaying the document.

Tip: You can use the MVAR statement in the template of an ODS list, ODS textblock, 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.

See: “Example 3: Creating a New Table Template ” on page 563 and “Example 1: Creating a Stand-Alone Style” on page 459

Syntax

```
MVAR variable-name-1 <='value-1' '<text-1>'>
< variable-name-2 <='value-2' '<text-2'>...>>;
```
Required Argument

variable-name

names a macro variable to reference in the template. ODS uses 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 or PROC ODSTABLE 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.

Optional Arguments

value

sets the default variable value.

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 converts 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.

Restriction:
The NMVAR statement can be used only in the template of an ODS list, ODS textblock, 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.

See: “Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

Syntax

NMVAR variable-name-1 <='value–1' <text-1'>>
< variable-name-2 <='value–2' <text-2'...>>;

Required Argument

variable-name

names a macro variable to reference in the template. ODS converts 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.
Optional Arguments

- **value**
  - sets the value of the variable.

- **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.

**Restriction:**
- The NOTES statement can be used only in the template of a table, column, header, 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.

**See:**
- “Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

**Example:**
- “Example 1: Creating a Customized Crosstabulation Table Template with No Legend” on page 390

---

Syntax

```
NOTES 'text';
```

**Required Argument**

- **text**
  - provides information about the table.

---

TEXT Statement

Specifies the text of a header, footer, or the label of a variable in an output data set.

**Restriction:**
- The TEXT statement can be used only within a header or footer template.

**See:**
- “Example 3: Creating a New Table Template” on page 563

---

Syntax

```
TEXT header/footer-specification(s);
```

**Required Argument**

- **header/footer-specification(s)**
  - specifies the text of the header or footer. Each header/footer-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 or footer is for a column, _LABEL_ specifies the label for the variable that is associated with the column. If the header or footer 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 or footer. 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 the SPLIT= option in the “DEFINE HEADER Statement” on page 532.

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 or footer specification, the header or footers a blank line.

Example
“Example 3: Creating a New Table Template ” on page 563

TEXT2 Statement

Provides an alternative header or footer to use in the LISTING output if the header or footer that is provided by the TEXT statement is too long.

Restriction: The TEXT2 statement can be used only within a header or footer template.

See: “TEXT Statement” on page 541

Syntax

TEXT2 header/footer-specification(s)

Required Argument

header/footer-specification(s)
specifies the text of the header or footer. Each header/footer-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 or footer is for a column, _LABEL_ specifies the label for the variable that is associated with the column. If the header or footer is for a table, _LABEL_ specifies the label for the data set that is associated with the table.
TEXT3 Statement

Provides an alternative header or footer to use in the LISTING output if the header or footer that is provided by the TEXT2 statement is too long.

Restriction: The TEXT3 statement can be used only within a header or footer template.

See: “TEXT Statement” on page 541

Syntax

TEXT3 header/footer-specification(s)

Required Argument

header/footer-specification(s)

specifies the text of the header or footer. Each header/footer-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 or footer is for a column, _LABEL_ specifies the label for the variable that is associated with the column. If the header or footer 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 or footer. 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 the SPLIT= option in the “DEFINE HEADER Statement” on page 532.
TRANSLATE INTO Statement

Translates the specified numeric values to other values.

Restrictions:

The TRANSLATE INTO statement can be used only in a column template, an ODS list, an ODS textblock, or a table template.

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.

Syntax

TRANSLATE expression-1 INTO expression-2 <, expression-n INTO expression-m>;

Required Arguments

expression-1

is an expression that is evaluated for each list item, paragraph, table, or column cell that contains a numeric variable.

If expression-1 resolves to TRUE (a nonzero 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.

equation has this form:

equation-1 <comparison-operator equation-n>

equation

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 about SAS functions, see SAS Functions and CALL Routines: Reference.

built-in variable

is a special type of WHERE expression operand that helps you find common values in table or column templates. Built-in variables are one or more of the following:

_COLUMN_

is a column number. Column numbering begins with 1.

Alias _COL_

Example “Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576
**_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.

Example  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

**_ROW_**

is a row number. Row numbering begins with 1.

Example  "Example 5: Setting the Style Element for a Specific Column, Row, and Cell" on page 576

**_STYLE_**

is a style element name.

Example  "Example 6: Creating Master Templates" on page 583

**_VAL_**

is the data value of a cell.

Tip  Use _VAL_ to represent the value of the current column.

Example  "Example 6: Creating Master Templates" on page 583

**comparison-operator**

compares a variable with a value or with another variable. The following table lists the comparison operators:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ^= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>

Table 15.4  Comparison Operators
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.

See  You can use any expression that can be used in the WHERE= data
      set option. For information about expressions that you can use in the
      WHERE data set option, see the WHERE data set option in SAS
      Data Set Options: Reference and the section on WHERE-Expression

Example  "Example 5: Setting the Style Element for a Specific Column, Row,
      and Cell" on page 576

`expression-2`

is an expression that specifies the value to use in the list, paragraph, or 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 about SAS functions, see SAS
Functions and CALL Routines: Reference.

`Built-in variable`

a special type 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_`

Example  "Example 5: Setting the Style Element for a Specific Column,
      Row, and Cell" on page 576

`_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
Example

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_ROW_

is a row number. Row numbering begins with 1.

Example

“Example 5: Setting the Style Element for a Specific Column, Row, and Cell” on page 576

_STYLE_

is a style element name.

See

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

Example

“Example 6: Creating Master Templates” on page 583

_VAL_

is the data value of a cell.

Tip

Use _VAL_ to represent the value of the current column.

Example

“Example 6: Creating Master Templates” on page 583

comparison-operator

compares a variable with a value or with another variable. The following table lists the comparison operators:

Table 15.5  Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or ~= or ~&lt; or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>

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 or 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 formatted field, starting at the left. To right-justify the value, use the JUSTIFY=ON attribute.
You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE data set option, see the WHERE data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.

Example 5: Setting the Style Element for a Specific Column, Row, and Cell

Usage: TEMPLATE Procedure: Creating Tabular Templates

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
.
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.

<table>
<thead>
<tr>
<th>just=left</th>
<th>just=center</th>
<th>just=right</th>
</tr>
</thead>
<tbody>
<tr>
<td>********</td>
<td>********</td>
<td>********</td>
</tr>
<tr>
<td>123.45</td>
<td>123.45</td>
<td>123.45</td>
</tr>
<tr>
<td>234.50</td>
<td>234.50</td>
<td>234.50</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>987.65</td>
<td>987.65</td>
<td>987.65</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>justify=on</th>
<th>justify=on</th>
<th>justify=on</th>
</tr>
</thead>
<tbody>
<tr>
<td>just=left</td>
<td>just=center</td>
<td>just=right</td>
</tr>
<tr>
<td>********</td>
<td>********</td>
<td>********</td>
</tr>
<tr>
<td>123.45</td>
<td>123.45</td>
<td>123.45</td>
</tr>
<tr>
<td>234.50</td>
<td>234.50</td>
<td>234.50</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>987.65</td>
<td>987.65</td>
<td>987.65</td>
</tr>
</tbody>
</table>

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 takes 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 are 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 15.6  Summary of Column Formatting Attributes

<table>
<thead>
<tr>
<th>Specifications Provided by the Column Template</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothing</td>
<td>Format name, width, and number of decimal places are determined by the data component.</td>
</tr>
<tr>
<td>Format name</td>
<td>Format name and width are determined by the column template; number of decimal places is determined by the data component.</td>
</tr>
<tr>
<td>Format name and width</td>
<td>Format name and width are determined by the column template.</td>
</tr>
<tr>
<td>Format name, width, and number of decimal places</td>
<td>All three are determined by the column template.</td>
</tr>
<tr>
<td>Width</td>
<td>No name is specified; width is determined by the column template; number of decimal places is determined by the data component.</td>
</tr>
<tr>
<td>Number of decimal places</td>
<td>No name is specified; width is determined by the data component; number of decimal places is determined by the column template.</td>
</tr>
</tbody>
</table>

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:
Examples: TEMPLAT E Procedure: Creating Tabular Templates

Example 1: Editing a Table Template That a SAS Procedure Uses

Features:
- EDIT statement
- Header attributes
  - JUST=
  - STYLE=
- Table attributes
  - DOUBLE_SPACE=
  - OVERLINE=
  - UNDERLINE=
- Other ODS features
  - ODS LISTING statement
  - ODS SELECT statement
  - DELETE statement

Data set: Exprev

Details

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.

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 “ODS HTML Statements for Running Examples in Different Operating Environments” in SAS Output Delivery System: User’s Guide.

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

```sas
options nodate pageno=1 pagesize=60 linesize=72;
ods listing;
ods select moments;
proc univariate data=exprev mu0=3.5;
  var Quantity;
  title "Default Moments Table";
run;
ods listing close;
```

Program Description

**Set the SAS system options.** The OPTIONS statement controls several aspects of the LISTING output.

```sas
options nodate pageno=1 pagesize=60 linesize=72;
```

**Create the LISTING output.** The ODS LISTING statement opens the LISTING destination and creates LISTING output.

```sas
ods listing;
```

**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 “ODS TRACE Statement” on page 76.

```sas
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.

```sas
proc univariate data=exprev mu0=3.5;
  var Quantity;
  title "Default Moments Table";
run;
```

**Stop the creation of the LISTING output.** The ODS LISTING CLOSE statement closes the LISTING destination and all the files that are associated with it. You must close the destination before you can view the output.

```sas
ods listing close;
```
Output from Program 1

Output 15.1 LISTING Output from PROC UNIVARIATE (Default Moments Table)

Output 15.2 HTML Output from PROC UNIVARIATE (Default Moments Table)

Program 2: Using a Customized Table Template

ods path sasuser.templat(update) sashelp.tmplmst(read);
proc template;
  edit base.univariate.moments;
    double_space=on;
    underline=on;
    overline=on;
    label="Custom Moments";
    double_space=on;
    style=data{color=orange fontstyle=italic};
  edit head;
    style=header{color=green fontstyle=italic};
    just=left;
end;
run;
ods listing;
ods select moments;
proc univariate data=exprev mu0=3.5;
  var Quantity;
  title "Custom Moments Table";
run;
ods listing close;
proc template;
delete base.univariate.moments;
end;
title;

Program Description

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. it creates a table template of the same name as the original one in Sashelp.tmplmst.

To learn the name of the table template that a procedure uses, run the procedure with the ODS TRACE ON statement in effect. For more information see "ODS TRACE Statement" on page 76.

proc template;
edit base.univariate.moments;

Specify changes to the Moments output object for the LISTING output. 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= creates 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;

Specify changes to the Moments output object for the HTML destination. These three table attributes affect the presentation of the Moments output object in the HTML output. DOUBLE_SPACE=ON specifies to double space between the rows of the table. The STYLE= statements specify a color and font style for the cell data. The LABEL= attribute changes the label that appears in the Results window from "Moments" to "Custom Moments".
Modify a table element. The following 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.HTMLBlue. Many procedures, including PROC UNIVARIATE, use this style element to produce headers for tables and columns. 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.

For information about viewing a style, see Chapter 19, "Style Templates," on page 727. For a table of style element names, see Chapter 20, "Style Elements," on page 797.

```
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 LISTING output. The ODS LISTING statement opens the LISTING destination and creates LISTING output.

```
ods listing;
```

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.

```
ods select moments;
```

Compute the descriptive statistics for one variable. PROC UNIVARIATE computes the univariate statistics for one variable, Quantity. 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.


```
proc univariate data=exprev mu0=3.5;
  var Quantity;
  title "Custom Moments Table";
run;
```
Stop the creation of the LISTING output. The ODS LISTING CLOSE statement closes the LISTING destination and all the files that are associated with it. You must close the destination before you can view the output.

    ods listing close;

Remove the customized moments table template from Sasuser.Templat. The DELETE statement removes the customized moments table that was created in this example. When using the DELETE statement, ODS looks for base.univariate.moments in Sasuser.Templat first. If it is there, it deletes it. If not, it searches Sashelp.Tmplmst.

    proc template;
    delete base.univariate.moments;
    end;
    title;

Output for Program 2

Output 15.3  LISTING Output (Customized Moments Table) from PROC UNIVARIATE

Output 15.4  Customized HTML Output (Customized Moments Table) from PROC UNIVARIATE (Viewed with Microsoft Internet Explorer)
Example 2: Comparing the EDIT Statement to the DEFINE TABLE Statement

Features:
- EDIT statement
- COLUMN statement
- DEFINE statement
  - STYLE= attribute
- NOTES statement
- DYNAMIC statement
- Other ODS features
  - ODS PATH statement
  - ODS HTML statement
  - DELETE statement

Data set: Exprev

Details
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 orange. 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=orange attribute.

Program 1

```sas
ods path work.templat (update) sashelp.tmpltmst (read);
proc template;
  edit Base.Summary;
  edit nobs;
  style={color=orange background=white};
end;
end;
run;

proc summary data=exprev print;
  class Sale_Type;
run;

proc template;
  delete base.Summary;
run;
```
Program Description

**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.

```ods path work.templat (update) sashelp.tmplmst (read);
proc template;
    edit Base.Summary;
    edit nobs;
    style={color=orange background=white};
end;
end;
run;
```

**Remove the customized summary table template from Work.Templat.** The DELETE statement removes the customized summary table that was created in this example. When using the DELETE statement, ODS looks for base.summary in Sasuser.Templat and Work.Templat first. If it is there, it deletes it. If not, it searches Sashelp.Tmplmst.

```proc template;
    delete base.Summary;
run;
```

**Output for Program 1**

The column labeled Sale_Type remains in the output because Sale_Type is defined as a dynamic variable, which is passed to the original Base.Summary table template, and Sale_Type is specified as the CLASS variable. The attributes of the NOBS column are modified in the EDIT statement where the NOBS column is defined.
The modified Base.Summary table template changes the foreground color of the NOBS column to orange. 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, submit odstemplates in the command bar. Then select Work.Templat ⇔ Base. Right-click the table template Summary and select Open. The table template Base.Summary is displayed in the Template Browser window.
Program 2

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 nobls id type ways (varname) (label) (min) (max) (range) (n) (nmiss) (sumwgt) (sum) (mean) (uss) (css) (var) (stddev) (cv) (stderr) (t) (probt) (lclm) (uc1m) (skew) (kurt) (median) (mode) (q1) (q3) (qrange) (p1) (p5) (p10) (p25) (p50) (p75) (p90) (p95) (p99);
  define nobls;
    style={color=orange backgroundcolor=white};
  end;
end;
run;

proc summary data=exprev print;
class Sale_Type;
run;

proc template;
delete base.Summary;
run;
**Program Description**

**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.

```plaintext
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=orange backgroundcolor=white};
    end;
end;
run;
```

**Remove the customized summary table template from Work.Templat.** The DELETE statement removes the customized summary table that was created in this example. When using the DELETE statement, ODS looks for base.summary in Sasuser.Templat and Work.Templat first. If it is there, it deletes it. If not, it searches Sashelp.Tmplmst.

```plaintext
proc template;
  delete base.Summary;
run;
```

**Output for Program 2**

The column labeled Sale_Type is missing because it was not defined in the new table template Base.Summary. The new table template only defined the NOBS column with an orange foreground and no column headers.
Output 15.7  HTML Output That Uses the Table Template Base.Summary.

The Base.Summary table template defines the foreground color of the NOBS column as orange. Because the vertical alignment and header 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, do the following: Submit `odstemplates` in the command bar. Then select `Work.Templat → Base`. Right-click the table template Summary and select `Open`. The table template Base.Summary is displayed in the Template Browser window.
Example 3: Creating a New Table Template

**Features:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table attributes</td>
<td></td>
</tr>
<tr>
<td>DOUBLE_SPACE=</td>
<td></td>
</tr>
<tr>
<td>OVERLINE=</td>
<td></td>
</tr>
<tr>
<td>UNDERLINE=</td>
<td></td>
</tr>
<tr>
<td>DEFINE statement (for columns)</td>
<td></td>
</tr>
<tr>
<td>GENERIC= attribute</td>
<td></td>
</tr>
<tr>
<td>HEADER= attribute</td>
<td></td>
</tr>
<tr>
<td>ID= attribute</td>
<td></td>
</tr>
<tr>
<td>STYLE= attribute</td>
<td></td>
</tr>
<tr>
<td>VJUST= attribute</td>
<td></td>
</tr>
<tr>
<td>DEFINE statement (for headers)</td>
<td></td>
</tr>
<tr>
<td>TEXT statement</td>
<td></td>
</tr>
<tr>
<td>STYLE= attribute</td>
<td></td>
</tr>
<tr>
<td>SPACE= attribute</td>
<td></td>
</tr>
<tr>
<td>DEFINE FOOTER statement</td>
<td></td>
</tr>
<tr>
<td>HEADER statement</td>
<td></td>
</tr>
<tr>
<td>MVAR statement</td>
<td></td>
</tr>
<tr>
<td>Other ODS features</td>
<td></td>
</tr>
<tr>
<td>FILE statement with ODS= option</td>
<td></td>
</tr>
<tr>
<td>PUT statement with <em>ODS</em> argument</td>
<td></td>
</tr>
</tbody>
</table>

**Data set:** Charity

**Details**

This example creates a custom table template for an output data set that PROC MEANS produces.
Program 1: Producing an Output Data Set with PROC MEANS

```sas
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;

proc print data=top3list noobs;
  title "Simple PROC PRINT of the Output Data Set";
run;
```

Program Description

Computes 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.

```sas
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 HTML output of the output data set that PROC MEANS created.

```sas
proc print data=top3list noobs;
  title "Simple PROC PRINT of the Output Data Set";
run;
```
Output 15.9  Default PROC PRINT Output

Program 2: Building a Custom Table Template for the TopN Report

    proc template;
    define table means.topn;
        mvar first_year last_year sysdate9;
        column class sum mean (raised) (name) (school) (year);
            double_space=on;
            overline=on;
            underline=on;
            header table_header_1 table_header_2;
    define table_header_1;
        text "Top Three Fund Raisers";
        style=header{fontsize=6};
    end;
    define table_header_2;
        text "from " first_year " to " last_year;
            space=1;
    end;
    define footer table_footer;
        text "(report generated on " sysdate9 ");
            split="*";
        style=header{fontsize=2};
    end;
    define class;
        generic=on;
        id=on;
        vjust=top;
        style=data;
    end;
    define sum;
        generic=on;
        header="Total Dollars Raised";
        vjust=top;
    end;

Example 3: Creating a New Table Template
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;
run;
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;
proc template;
delete means.topn;
run;

Program Description

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= creates 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;
  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.HTMLBlue. 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};
```
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. 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, Styles.Default. 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;
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.

```plaintext
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.

```plaintext
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.

```plaintext
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. 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;

Remove the customized means table template. The DELETE statement removes the customized means table that was created in this example. When using the DELETE statement, ODS looks for means.topn in Sasuser.Templat and Work.Templat first. If it is there, it will delete it. If not, it will search Sashelp.Tmplmst.

   proc template;
   delete means.topn;
   run;
Example 4: Setting the Style Element for Cells Based on Their Values

**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
  - `DELETE` statement
  - `FILE` statement with `ODS=` option
  - `PUT` statement with `_ODS_` argument

**Data set:** Grain_Production

**Format:** $CNTRY.
Details

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 “ODS HTML Statements for Running Examples in Different Operating Environments” in SAS Output Delivery System: User’s Guide.

Program

```sas
program
  title "Leading Grain Producers";
  proc template;
    define table shared.cellstyle;
      translate _val_=. into "No data";
      notes "NMVAR defines symbols that will be used to determine the colors of the cells.;"
      nmvar low "Use default style."
        medium "Use yellow foreground color and bold font weight"
          high "Use red foreground color and a bold, italic font.;"
      classlevels=on;
    define column char_var;
      generic=on;
      blank_dups=on;
      end;
    define column num_var;
      generic=on;
      justify=on;
      %let low=10000;
      %let medium=50000;
      %let high=100000;
      cellstyle _val_ <= &low as data,
        _val_ <= &medium as data
          {color=green fontstyle=italic},
        _val_ <= &high as data
          1 as data
          {color=yellow fontweight=bold},
      end;
      end;
    run;
  data _null_;
    set grain_production;
```

file print ods=(
    template="shared.cellstyle"
    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.)
    )
);

put _ods_;
run;

proc template;
    delete shared.cellstyle;
run;

Program Description

Specify a title. The TITLE statement specifies a title.

    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.

```plaintext
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.

```plaintext
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.

```plaintext
justify=on;
```

Assign values to three macro variables. The %LET statements assign values to the macro variables LOW, MEDIUM, and HIGH.

```plaintext
%let low=10000;
%let medium=50000;
%let high=100000;
```

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 END statement ends the column template.

```plaintext
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.

```plaintext
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 Grain_Production.

```sas
data _null_
  set grain_production;
run;
```

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. The TEMPLATE= suboption tells ODS to use the table template named Shared.CellStyle, which was previously created with PROC TEMPLATE.

```sas
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.

```sas
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.

```sas
put _ods_; 
run;
```

Remove the customized table template. The DELETE statement removes the customized table that was created in this example. When using the DELETE statement, ODS looks for shared.cellstyle in Sasuser.Templat and Work.Templat first. If it is there, it will delete it. If not, it will search Sashelp.Tmplmst.

```sas
proc template;
  delete shared.cellstyle;
run;
```

HTML Output of a Customized Table

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.
Example 5: Setting the Style Element for a Specific Column, Row, and Cell

Features:
- DEFINE STYLE statement
- REPLACE statement
- DEFINE TABLE statement
- CELLSTYLE AS statement
- DEFINE COLUMN statement
- DEFINE HEADER statement: TEXT statement
Details

This example combines a customized style with a customized table template to produce output with a checkerboard pattern of table cells.

Program

```plaintext
proc template;
    define style greenbar;
        parent=styles.printer;
        replace headersandfooters from cell /
            backgroundcolor=light green
color=black
        fontsize=3
        fontweight=bold
    ;
end;
run;
ods html body="greenbar.html" style=greenbar;
ods pdf file="greenbar.pdf" style=greenbar;
ods trace on;
proc template;
    define table Checkerboard;
        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 fontweight=bold },
1 as data;
define header top;
    text "Checkerboard Table Template";
end;
    define column country;
        dataname=country;
    define header bar;
        text "Country";
    end;
    header=bar;
end;
    define column OrderDate;
```
Program Description

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 PDF output and specify the style that you want to use for the output. The ODS HTML statement sends all output objects to the file greenbar.html. 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;
ods trace on;

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. Similarly, if both the row and column are even numbered, then the cell is yellow.

    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 fontweight=bold },
    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;
    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 template 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.

```plaintext
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. The TEMPLATE= suboption tells ODS to use the table template named Checkerboard.

```plaintext
file print ods=(template="Checkerboard");  
  put _ods_;  
run;
```

Stop the creation of PDF output. The ODS HTML CLOSE 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 PDF destination before you can view the output.

```plaintext
ods html close;  
ods pdf close;  
ods html;
```
### Output

**Output 15.12**  HTML Output (Viewed with Internet Explorer 6.0)

<table>
<thead>
<tr>
<th>Country</th>
<th>Order Date</th>
<th>Ship Date</th>
<th>Sale Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antarctica</td>
<td>1/1/05</td>
<td>1/7/05</td>
<td>Internet</td>
</tr>
<tr>
<td>Puerto Rico</td>
<td>1/1/05</td>
<td>1/3/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Virgin Islands (U.S.)</td>
<td>1/1/05</td>
<td>1/4/05</td>
<td>In Store</td>
</tr>
<tr>
<td>Aruba</td>
<td>1/1/05</td>
<td>1/4/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Bahamas</td>
<td>1/1/05</td>
<td>1/4/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Bermuda</td>
<td>1/1/05</td>
<td>1/4/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Belize</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>In Store</td>
</tr>
<tr>
<td>British Virgin Islands</td>
<td>1/2/05</td>
<td>1/3/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Canada</td>
<td>1/2/05</td>
<td>1/5/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Cayman Islands</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>In Store</td>
</tr>
<tr>
<td>Costa Rica</td>
<td>1/2/05</td>
<td>1/6/05</td>
<td>Internet</td>
</tr>
<tr>
<td>Cuba</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>Internet</td>
</tr>
<tr>
<td>Dominican Republic</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>Internet</td>
</tr>
<tr>
<td>El Salvador</td>
<td>1/2/05</td>
<td>1/6/05</td>
<td>Catalog</td>
</tr>
<tr>
<td>Guatemala</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>In Store</td>
</tr>
<tr>
<td>Haiti</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>Internet</td>
</tr>
<tr>
<td>Honduras</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>Internet</td>
</tr>
<tr>
<td>Jamaica</td>
<td>1/2/05</td>
<td>1/4/05</td>
<td>In Store</td>
</tr>
<tr>
<td>Mexico</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>In Store</td>
</tr>
<tr>
<td>Montserrat</td>
<td>1/2/05</td>
<td>1/2/05</td>
<td>In Store</td>
</tr>
</tbody>
</table>
### Example 6: Creating Master Templates

**Features:**
- DEFINE TABLE statement
  - CELLSTYLE AS statement: STYLE\_variable
  - CELLSTYLE AS statement: _ROW_ variable
- DEFINE COLUMN statement
  - CELLSTYLE AS statement: _VAL_ variable
  - STYLE= column attribute
- DEFINE HEADER statement
Details
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

   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;
      define column base.template.column;
          style={fontstyle=italic};
          cellstyle _val_ > 5 as {fontsize=15pt},
          _val_ = "Num" as {fontsize=20pt};
      end;
      define header base.template.header;
          style={fontsize=20pt color=purple};
      end;
      link base.template.footer to base.template.header;
      run;
   ods select variables;
   proc contents data=sashelp.class;
   run;
   proc template;
      delete base.template.table;
      delete base.template.column;
      delete base.template.header;
      delete base.template.footer;
      run;

Program Description
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. This template 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. This template 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 specifies that column fonts are italicized. The first CELLSTYLE-AS statement specifies that if the value of the cell is greater than five, then 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 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;
```

View the contents of the SAS data set. The CONTENTS procedure shows the contents of the SAS data set Sashelp.Class.

```
ods select variables;
proc contents data=sashelp.class;
run;
```

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

Output 15.14  Using Master Templates for HTML Output

Example 7: Table Header and Footer Border Formatting

Features:
- Border control style attributes
  - BORDERBOTTOMCOLOR=
  - BORDERBOTTOMSTYLE=
  - BORDERBOTTOMWIDTH=
  - BORDERTOPCOLOR=
  - BORDERTOPSTYLE=
  - BORDERTOPWIDTH=
- DEFINE statement
- DEFINE STYLE statement
- EDIT statement
- FOOTER statement
- HEADER statement
Details

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 valid only in the RTF destination.

Program

```plaintext
options nodate nonumber;
title "TableHeaderContainer, TableFooterContainer, and Border Control Style"
    " Attributes";
title2 "Allows Control of Borders Between the Header, Body, and Footer of a"
    " Table";
ods html close;
proc template;
    define style  HeadersFootersBorders;
        parent=styles.rtf;
    style TableHeaderContainer from TableHeaderContainer /
        borderbottomwidth=12
        borderbottomcolor=blue
        borderbottomstyle=dotted;
    style TableFooterContainer from TableFooterContainer /
        bordertopwidth=6
        bordertopcolor=red
        bordertopstyle=double;
    style table from table /
        cellspacing=0 rules=groups frame=void;
end;
run;
proc template;
    edit Base.Datasets.Members;
        header hd1;
        footer ft1;
        define hd1;
            preformatted=on;
            just=1;
            text"     Table Header with Leading and Trailing Blanks    ";
```
Program Description

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

options nodate nonumber;
title "TableHeaderContainer, TableFooterContainer, and Border Control Style" " Attributes";
title2 "Allows Control of Borders Between the Header, Body, and Footer of a" " Table";

Close the HTML destination. The ODS HTML CLOSE statement closes the HTML destination to conserve system resources.

ods html close;

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
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 CELLSspacing=, 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.

```
proc template;
   edit Base.Datasets.Members;
      header hd1;
      footer ft1;
      define hd1;
         preformatted=on;
         just=1;
         text "     Table Header with Leading and Trailing Blanks    ";
      end;
      define ft1;
         preformatted=on;
         just=1;
         text "     Table Footer with Leading and Trailing Blanks    ";
      end;
      edit name;
      define header myheader;
         just=1;
         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=work;
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 rtf close;

**RTF Output**

*Output 15.15  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>
<thead>
<tr>
<th>#</th>
<th>My new header</th>
<th>Type</th>
<th>File Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STATS</td>
<td>DATA</td>
<td>5120</td>
<td>19Oct10:14:31:34</td>
</tr>
<tr>
<td>2</td>
<td>STATS</td>
<td>DATA</td>
<td>5120</td>
<td>19Oct10:14:31:34</td>
</tr>
</tbody>
</table>

---
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 15, “TEMPLATE Procedure,” on page 515.

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 about table attributes, see “Table Attributes” on page 620.
Column Attributes

This section lists all the attributes that you can use in a column template. Column attributes are used to customize the attributes of a column and must be specified within a DEFINE COLUMN statement block. You can specify multiple column attributes together or separately. For example, you can specify the following column attributes together:

```
blank_dups=yes data_format_override=off just=l vjust=r;
```

or separately:

```
blank_dups=yes;
data_format_override=OFF;
just=l vjust=r;
```

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 16.1  Boolean Values**

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td><em>ON</em></td>
<td><em>OFF</em></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><em>YES</em></td>
<td><em>NO</em></td>
</tr>
</tbody>
</table>

**Table 16.2  Column Attributes**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Influence the appearance of the cells contents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether to suppress the value of a variable from</td>
<td>BLANK_DUPS= on</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>page 596</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>----------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>one row to the next, if the value does not change based on the</td>
<td>BLANK_INTERNAL_DU</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>formatted value of the variable</td>
<td>PS= on page 596</td>
<td></td>
</tr>
<tr>
<td>Specify whether to suppress the value of a variable from one row</td>
<td>CHOOSE_FORMAT= on page</td>
<td>All</td>
</tr>
<tr>
<td>to the next if the value does not change based on the raw value of</td>
<td>597</td>
<td></td>
</tr>
<tr>
<td>the variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select the best format for a column of a table</td>
<td>FLOW= on page 598</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to wrap the text in the current column</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the format for the column</td>
<td>FORMAT= on page 598</td>
<td>All</td>
</tr>
<tr>
<td>Specify the number of decimals for the column if it is not specified</td>
<td>FORMAT_NDEC= on page</td>
<td>All</td>
</tr>
<tr>
<td>with FORMAT= column attribute</td>
<td>599</td>
<td></td>
</tr>
<tr>
<td>Specify the format width for the column if it is not specified with</td>
<td>FORMAT_WIDTH= on page</td>
<td>All</td>
</tr>
<tr>
<td>FORMAT= column attribute</td>
<td>599</td>
<td></td>
</tr>
<tr>
<td>Supply a numeric value against which values in the column are</td>
<td>FUZZ= on page 599</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>compared to eliminate trivial values from printing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the horizontal justification of the format field within</td>
<td>JUST= on page 601</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>the column (and for the column header if the template for the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header does not include JUST=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether to justify the format field within the column, or</td>
<td>JUSTIFY= on page 602</td>
<td>All destinations except</td>
</tr>
<tr>
<td>to justify the value within the column, without regard to the</td>
<td></td>
<td>LISTING behave as if</td>
</tr>
<tr>
<td>format field</td>
<td></td>
<td>JUSTIFY=ON</td>
</tr>
<tr>
<td>When the text in the column uses more than one line, specify</td>
<td>MAXIMIZE= on page 602</td>
<td>LISTING</td>
</tr>
<tr>
<td>whether to try to divide the text equally among all lines or to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maximize the amount of text in each line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>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</td>
<td>OVERLINE= on page 603</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to treat the text as preformatted text</td>
<td>PREFORMATTED= on page 604</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>Specify whether to print the column</td>
<td>PRINT= on page 605</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify a separator character to append to each value in the column</td>
<td>SEPARATOR= on page 605</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the style element and style attributes to use for the column</td>
<td>STYLE= on page 606</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>Specify that the text graphic columns be turned off when a procedure is going to output a graph</td>
<td>TEXT_GRAPHIC= on page 607</td>
<td>All except OUTPUT and DOCUMENT</td>
</tr>
<tr>
<td>Specify the split character for the data in the column</td>
<td>TEXT_SPLIT= on page 607</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>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</td>
<td>UNDERLINE= on page 607</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the vertical justification for the column</td>
<td>VJUST= on page 608</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>Specify the width of the column in characters</td>
<td>WIDTH= on page 608</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the maximum width for this column</td>
<td>WIDTH_MAX= on page 608</td>
<td>LISTING</td>
</tr>
<tr>
<td>Customize column headers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the text for the column header or the name of the header template</td>
<td>HEADER= on page 600</td>
<td>All</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>Specify whether to print the column header</td>
<td>PRINT_HEADERS= on page 605</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Influence the relationship to other columns</td>
<td>GENERIC= on page 599</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether the column template is generic (that is, whether more than one variable use the template)</td>
<td>ID= on page 601</td>
<td>LISTING and printer family</td>
</tr>
<tr>
<td>Specify whether the column is an ID column</td>
<td>MERGE= on page 603</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to merge the current column with the column immediately to its right</td>
<td>PRE_MERGE= on page 604</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to merge the current column with the column immediately to its left</td>
<td>PRE_SPACE= on page 604</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the number of blank characters to leave between the current column and the column immediately to its left</td>
<td>SPACE= on page 605</td>
<td>LISTING</td>
</tr>
<tr>
<td>Influence the presentation of data panels</td>
<td>GLUE= on page 600</td>
<td>LISTING, printer family, and RTF</td>
</tr>
<tr>
<td>Influence the place at which ODS splits a table when it creates multiple data panels</td>
<td>OPTIONAL= on page 603</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify which format to use if both a column template and a data component specify a format</td>
<td>DATA_FORMAT_OVER_RIDE= on page 598</td>
<td>All</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Specify the name of the column in the data component to associate with the current column</td>
<td>DATANAME= on page 598</td>
<td>All</td>
</tr>
<tr>
<td>Specify which special characters in headers for generic columns are to be used as split characters</td>
<td>DEF_SPLIT on page 598</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether to include the column in an output data set</td>
<td>DROP= on page 598</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>Specify a label for the column</td>
<td>LABEL= on page 602</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>Specify the column template that the current template inherits from</td>
<td>PAREN= on page 604</td>
<td>All</td>
</tr>
<tr>
<td>Specify the name to use for the corresponding variable in an output data set</td>
<td>VARNAME= “VARNAME=variable-name</td>
<td>variable;” on page 607</td>
</tr>
</tbody>
</table>

**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 is in effect, then ODS ignores BLANK_DUPS=ON when any value changes in a preceding column that is also marked with BLANK_DUPS=ON.

**Tips**
The BLANK_DUPS attribute is valid in all destinations except the OUTPUT destination.

When the PRINTER destination suppresses the value of a variable, it also suppresses the horizontal rule above the blank cell.

**See**
CLASSLEVELS= table attribute on page 625

**Example**
“Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

**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 is in effect, then ODS ignores BLANK_INTERNAL_DUPS=ON when any value changes in a
preceding column that is also marked with BLANK_INTERNAL_DUPS=ON.

Tips

The BLANK_INTERNAL_DUPS attribute is valid in all destinations except the OUTPUT destination.

When the PRINTER destination suppresses the value of a variable, it also suppresses the horizontal rule above the blank cell.

See

CLASSLEVELS= table attribute on page 625

**CHOICE_FORMAT= COMPROMISE | MAX | MAX_ABS | MIN_MAX;**

selects a format based on the actual values in the column of the table.

**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.

Tips

FORMAT_NDEC=d specifies the precision in digits.

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.

Default

If you omit the CHOICE_FORMAT column attribute, then the default format is determined by either the data component or other attributes.

Restriction

CHOICE_FORMAT is not supported for computed columns because those columns' values are computed outside of the data object.

Tips

If you specify a small value for the FORMAT_WIDTH= option, then CHOICE_FORMAT might create a dw.3 format.

The CHOICE_FORMAT= attribute is valid in all destinations.

See

For more information about column formats, see "Formatting Values in Table Columns" on page 549.
**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 in the data component.

**OFF**
uses the format in the column template.

**variable**
uses the format of the specified variable.

Default OFF

Tip The DATA_FORMAT_OVERRIDE attribute is valid in all destinations.

**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 that 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.

Tips The FLOW attribute is valid only in the LISTING destination.

The HTML and PRINTER destinations always wrap the text if it is too long to fit in the space that is provided.

See MAXIMIZE= on page 602

**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, or the length of the variable for character variables.

<table>
<thead>
<tr>
<th>Restriction</th>
<th>If you specify a format width for a numeric column, then its value cannot exceed 32.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>The FORMAT= attribute is valid in all destinations.</td>
</tr>
</tbody>
</table>

**FORMAT_NDEC= number | variable;**
specifies the number of decimals for the column.

<table>
<thead>
<tr>
<th>Default</th>
<th>The decimal width that is specified with the FORMAT= column attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>Number is a whole number from 0 to 32</td>
</tr>
<tr>
<td>Interaction</td>
<td>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.</td>
</tr>
<tr>
<td>Tip</td>
<td>The FORMAT_NDEC= attribute is valid in all attributes.</td>
</tr>
</tbody>
</table>

**FORMAT_WIDTH=positive-integer | variable;**
specifies the format width for the column.

<table>
<thead>
<tr>
<th>Default</th>
<th>If you omit the column attribute FORMAT_WIDTH=, then ODS uses the format specified in the FORMAT= attribute.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>1 to 32 for numeric variables; operating system limit for character variables</td>
</tr>
<tr>
<td>Interaction</td>
<td>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.</td>
</tr>
<tr>
<td>Tip</td>
<td>The FORMAT_WIDTH= attribute is valid in all destinations.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Default</th>
<th>This is the smallest representable floating-point number on the computer that you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>The FUZZ= attribute is valid in all destinations except the OUTPUT destination.</td>
</tr>
</tbody>
</table>

**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                                                                                               |
### 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
- **Tips**: A value of -1 forces the table to split between the current column and the column to its right.

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 in 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 532). 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.

- **Tips**: 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 532.)

- Use the split character in the text of the header to force the text to a new line.
The HEADER= attribute is valid in all destinations.

### See

- "LABEL=" text | variable;" on page 602 and
- "TEXT_SPLIT=" character" | variable;" on page 607

### Examples

- “Example 3: Creating a New Table Template ” on page 563
- “Example 1: Creating a Stand-Alone Style” on page 459

### 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

**Tips**

ODS treats all columns up to and including a column that is marked with ID=ON as ID columns.

The ID attribute is valid only in LISTING and printer family destinations.

**Example**

“Example 3: Creating a New Table Template ” on page 563

### 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.
Interactions
The TEXTALIGN= style attribute on page 886 overrides the value of JUST=.

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 602.

Tip
The JUST= attribute is valid in all destinations except the OUTPUT destination.

See
"Values in Table Columns and How They Are Justified" on page 548

Example
“Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551

JUSTIFY<=ON | OFF | variable>; indicates 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.

Tips
If you translate numeric data to character data, you might need to use JUSTIFY= to align the data.

All destinations except the LISTING destinations justify values as if JUSTIFY=ON.

See
"Values in Table Columns and How They Are Justified" on page 548

Example
“Example 4: Setting the Style Element for Cells Based on Their Values” on page 571

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.

Tips
The LABEL= attribute is valid only in the OUTPUT destination.

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.

<table>
<thead>
<tr>
<th>Default</th>
<th>OFF</th>
</tr>
</thead>
</table>

**Interaction**  
This attribute is effective only if the column is defined with FLOW=ON. (See the discussion of the FLOW= column attribute on page 598).

**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.

<table>
<thead>
<tr>
<th>Default</th>
<th>OFF</th>
</tr>
</thead>
</table>

**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**  
PRE_MERGE= column attribute on page 604

**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.

<table>
<thead>
<tr>
<th>Default</th>
<th>OFF</th>
</tr>
</thead>
</table>

** 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.

<table>
<thead>
<tr>
<th>Default</th>
<th>OFF</th>
</tr>
</thead>
</table>

**Tip**  
The OVERLINE= attribute is valid only in the LISTING destination.

**See**  
For information about formatting characters, see the FORMCHAR= table attribute on page 627.
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  MERGE= column attribute on page 603

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=.

Tip The PRE_SPACE= attribute is valid only in the LISTING destination.

See SPACE= column attribute on page 605, COL_SPACE_MIN table attribute on page 625, and COL_SPACE_MIN table attribute on page 625.

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.

Tips If all columns in a stacked column have PRINT=OFF specified, then the entire column is removed from the table. The PRINT attribute is valid in all destination except the OUTPUT destination.

See OPTIONAL column attribute on page 603 and DROP= column attribute on page 598.

PRINT_HEADERS<=ON | OFF | variable>; specifies whether to print the column header and any underlining and overlining.
Default ON

Tip The PRINT_HEADERS attribute is valid in all destination except the OUTPUT destination.

See UNDERLINE= column attribute on page 607 and OVERLINE= column attribute on page 603.

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.

Tips To specify a hexadecimal character as the separator character, put an x after the closing quotation mark. For example, this option assigns the hexadecimal character 2-D as the separator character: separator="2D"x

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.
A value in the range that is bounded by the COL_SPACE_MIN and COL_SPACE_MAX table attributes.

If PRE_SPACE= and SPACE= are specified for the same intercolumn space, ODS honors PRE_SPACE=.

The SPACE= attribute is valid only in the LISTING destination.

The PRE_SPACE= column attribute on page 604, the COL_SPACE_MAX= attribute on page 625, and the “COL_SPACE_MIN= positive-integer | variable;” on page 625.

**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 14, “TEMPLATE Procedure,” on page 425).

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
- DataEmphasisFixed
- DataStrong
- DataStrongFixed

The style element provides the basis for displaying the column. Additional style attributes that you provide can modify the display.

For information about viewing a style so that you can see the style elements that are available, see “Viewing the Contents of a Style” on page 429. For information about the default style that ODS uses, see “Modifying Style Elements in the Default Style for HTML and Markup Languages” on page 431.

*style-element-name* is either the name of a style element or a variable whose value is a style element. For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

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 about the style attributes that you can specify, see “About Style Attributes” on page 459.

**Tips**
The STYLE= attribute is valid only in the markup family, printer family, and RTF destinations.

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.

**Example**

“Example 3: Creating a New Table Template ” on page 563

**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

Tip The UNDERLINE= attribute is valid only in the LISTING destination.

See For information about formatting characters, see the FORMCHAR= table attribute on page 627.

**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.

Tips 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.

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.
Example “Example 3: Creating a New Table Template " on page 563

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, or data length for character variables.
Interaction The length of the column header can influence the width of the column.
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.

Header and Footer Attributes
This section lists all the attributes that you can use in a header or footer template. Header and footer attributes are used to customize the attributes of a header or footer. They can be specified only within a DEFINE HEADER statement block or a
DEFINE FOOTER statement block. You can specify multiple attributes together or separately. For example, you can specify the following attributes together:

overline=on underline=on just=c;

or separately:

overline=on;
underline=on;
just=c;

A column header spans a single column. A spanning header spans multiple columns. These two types 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>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td><em>ON</em></td>
<td><em>OFF</em></td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><em>YES</em></td>
<td><em>NO</em></td>
</tr>
</tbody>
</table>

For all 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 16.3  Boolean Values

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Influence the appearance of the contents of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>the header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify that special characters</td>
<td>DEF_SPLIT on page 598</td>
<td>All</td>
</tr>
<tr>
<td>in headers for generic columns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>are to be used as split</td>
<td></td>
<td></td>
</tr>
<tr>
<td>characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify a character to use to</td>
<td>EXPAND= on page 612</td>
<td>LISTING</td>
</tr>
<tr>
<td>expand the header to fill the space over the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>column or columns that the header spans</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>----------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Specify whether to try to expand the column width to accommodate</td>
<td>FORCE= on page 613</td>
<td>LISTING</td>
</tr>
<tr>
<td>the longest word in the column header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the horizontal justification for the column header</td>
<td>JUST= on page 614</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to try to divide the text equally among all lines</td>
<td>MAXIMIZE= on page 614</td>
<td>LISTING</td>
</tr>
<tr>
<td>or to maximize the amount of text in each line when the text in the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header uses more than one line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether to draw a continuous line above the header</td>
<td>OVERLINE= on page 615</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to treat the text as preformatted text</td>
<td>PREFORMATTED= on page 615</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>Specify whether to print the header</td>
<td>PRINT= on page 605</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether to repeat the text of the header until the space</td>
<td>REPEAT= on page 615</td>
<td>LISTING</td>
</tr>
<tr>
<td>that is allotted for the header is filled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the number of blank lines to place between the current</td>
<td>SPACE= on page 616</td>
<td>LISTING</td>
</tr>
<tr>
<td>header and the next header or between the current footer and the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>previous footer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the split character for the header</td>
<td>SPLIT= on page 617</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify the style element and any changes to its attributes to use</td>
<td>STYLE= on page 606</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>for the header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether to start a new header line in the middle of a word</td>
<td>TRUNCATE= on page 619</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to draw a continuous line underneath the header</td>
<td>UNDERLINE= on page 607</td>
<td>LISTING</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Specify vertical justification for the header</td>
<td>VJUST= on page 619</td>
<td>Markup family, PRINTER, family, and RTF</td>
</tr>
<tr>
<td>Specify the width of the header in characters</td>
<td>WIDTH= on page 608</td>
<td>LISTING</td>
</tr>
<tr>
<td>Influence the content of the header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Influence the placement of the header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the last column that a spanning header covers</td>
<td>END= on page 612</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to expand the header to reach the sides of the page</td>
<td>EXPAND_PAGE= on page 613</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether a spanning header appears only on the first data</td>
<td>FIRST_PANEL= on page 613</td>
<td>LISTING, printer family, and RTF</td>
</tr>
<tr>
<td>panel if the table is too wide to fit in the space that is provided</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether a table footer appears only on the last data panel</td>
<td>LAST_PANEL= on page 614</td>
<td>LISTING, printer family, and RTF</td>
</tr>
<tr>
<td>if the table is too wide to fit in the space that is provided</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether to extend the text of the header into the header</td>
<td>SPILL_ADJ= on page 616</td>
<td>LISTING</td>
</tr>
<tr>
<td>space of adjacent columns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the first column that a spanning header covers</td>
<td>START= on page 617</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to extend the text of the header into the adjacent</td>
<td>SPILL_MARGIN= on page 616</td>
<td>LISTING</td>
</tr>
<tr>
<td>margin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other header attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify an abbreviation for the header</td>
<td>ABBR= on page 612</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Specify an acronym for the header</td>
<td>ACRONYM= on page 612</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Destinations</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>Specify an alternate description for the header</td>
<td>ALT= on page 612</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Specify whether multiple columns can use the header</td>
<td>GENERIC= on page 613</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify a long description for the header</td>
<td>LONGDESC= on page 614</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Specify the header template that the current template inherits from</td>
<td>PARENT= on page 615</td>
<td>All</td>
</tr>
</tbody>
</table>

**ABBR= "text" | variable;**
specifies an abbreviation for the header.

- **Requirement**
  The text must be enclosed in quotation marks.

- **Tip**
  The ABBR attribute is valid only in the MARKUP destination.

**ACRONYM= "text" | variable;**
specifies an acronym for the header.

- **Requirement**
  The text must be enclosed in quotation marks.

- **Tip**
  The ACRONYM= attribute is valid only in the MARKUP destination.

**ALT= "text" | variable;**
specifies an alternate description of the header.

- **Requirement**
  The text must be enclosed in quotation marks.

- **Tip**
  The ALT= attribute is valid only in the MARKUP destination.

**DEF_SPLIT;**
specifies that 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

- **Tip**
  The END= attribute is valid in all destinations except the OUTPUT destination.

- **See**
  START= header attribute on page 617

**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.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Interaction</th>
<th>Tips</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPAND_PAGE</td>
<td>OFF</td>
<td>Specifies whether to expand the header to reach the sides of the page.</td>
<td>If the string or the variable that you specify contains more than one character, then only the first character is used. The EXPAND= attribute is valid only in the LISTING destination.</td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>ON</td>
<td>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.</td>
<td>The FORCE= attribute is valid only in the LISTING destination.</td>
<td>WIDTH= column attribute on page 608 and WIDTH_MAX= column attribute on page 608.</td>
</tr>
<tr>
<td>Generic</td>
<td>OFF</td>
<td>Specifies whether multiple columns can use the header.</td>
<td>This attribute is primarily for writers of SAS procedures and for DATA step programmers.</td>
<td></td>
</tr>
<tr>
<td>FIRST_PANEL</td>
<td>OFF</td>
<td>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.</td>
<td>The FIRST_PANEL attribute is valid in the LISTING, printer family, and RTF destinations.</td>
<td>LAST_PANEL header attribute on page 614</td>
</tr>
<tr>
<td>GENERIC</td>
<td>OFF</td>
<td>Specifies whether multiple columns can use the header.</td>
<td>This attribute is primarily for writers of SAS procedures and for DATA step programmers.</td>
<td></td>
</tr>
</tbody>
</table>
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.

Example "Example 1: Editing a Table Template That a SAS Procedure Uses" on page 551

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

Tip The LAST_PANEL= attribute is valid only in the LISTING, printer family, and RTF destinations.

See FIRST_PANEL= header attribute on page 613

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.

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 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 “FORMCHAR= "string" | variable;” on page 627.

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

**Interactions**

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

**Tips**

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.

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.
### 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

**Tips**
- A row of underlining or overlining is considered a header or a footer.
- The SPACE= attribute is valid only in the LISTING destination.

#### Example

"Example 3: Creating a New Table Template " on page 563

### 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=.

**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=.

**Tip**
- The SPILL_MARGIN attribute is valid only in the LISTING destination.

**See**
- FORCE= header attribute on page 613, SPILL_MARGIN= on page 616, and TRUNCATE= header attribute on page 619.
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.

The first character in a header defines the split character if it is not one of the following:

- an alphanumeric character
- a blank
- an underscore (_)
- a hyphen (-)
- a period (.)
- a percent (%)

Note: The split will not occur until the split character appears after text.

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

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 448).

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**

- “Viewing the Contents of a Style” on page 429
- “Finding and Viewing the Default Style for ODS Destinations” on page 429
- For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**style-attribut-specification**

describes the style attribute to change. Each `style-attribut-specification` has this general form:

`style-attribut-name=style-attribut-value`

**Requirement**

The STYLE= option requires either a `style-attribut-specification` or a `style-element-name`.

**Tips**

- You can use braces ({ and }) instead of square brackets ([ and ]). 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.
The STYLE= attribute is valid only in the markup family, printer family, and RTF destinations.

See Chapter 21, "Style Attributes," on page 827

Examples

"Example 1: Editing a Table Template That a SAS Procedure Uses" on page 551

"Example 3: Creating a New Table Template " on page 563

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. 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.

Tip The TRUNCATE= attribute is valid only in the LISTING destination.

See FORCE= header attribute on page 613, SPILL_MARGIN= header attribute on page 616, and SPILL_ADJ header attribute on page 616.

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

Tip The UNDERLINE attribute is valid only in the LISTING destination.

See For information about formatting characters, see the discussion of FORMCHAR= table attribute on page 627.

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
Table Attributes

This section lists all the attributes that you can use in a table template. Table attributes are used to customize the attributes of a table and must be specified within a DEFINE TABLE statement block. You can specify multiple table attributes together or separately. For example, you can specify the following table attributes together:

```
order_data=yes use_format_defaults=yes print_headers=off;
```

or separately:

```
order_data=yes;
use_format_defaults=yes;
print_headers=off;
```

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>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td><em>ON</em></td>
<td><em>OFF</em></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>Influence the layout of the table</td>
<td></td>
</tr>
<tr>
<td>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</td>
<td>BALANCE on page 624</td>
</tr>
<tr>
<td>Specify whether to center each data panel independently if the entire table does not fit in one data panel</td>
<td>CENTER on page 624</td>
</tr>
<tr>
<td>Specify whether to force a new page before printing the table</td>
<td>NEWPAGE on page 628</td>
</tr>
<tr>
<td>Specify the number of sets of columns to place on a page</td>
<td>PANELS= on page 629</td>
</tr>
<tr>
<td>Specify the number of blank characters to place between sets of columns when PANELS= is in effect</td>
<td>PANEL_SPACE= on page 629</td>
</tr>
<tr>
<td>Specify the number of lines that must be available on the page in order to print the body of the table</td>
<td>REQUIRED_SPACE= on page 630</td>
</tr>
<tr>
<td>Specify the number of lines to place between the previous output object and the current one</td>
<td>TOP_SPACE= on page 631</td>
</tr>
<tr>
<td>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</td>
<td>WRAP on page 632</td>
</tr>
<tr>
<td>Specify whether to add a double space after the last line of a single row when the row is wrapped</td>
<td>WRAP_SPACE on page 632</td>
</tr>
<tr>
<td>Influence the layout of rows and columns</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>Specify the maximum number of blank characters to place between columns</td>
<td>COL_SPACE_MAX= on page 625</td>
</tr>
<tr>
<td>Specify the minimum number of blank characters to place between columns</td>
<td>COL_SPACE_MIN= on page 625</td>
</tr>
<tr>
<td>Specify the name of the column whose value provides formatting information about the space before each row of the template</td>
<td>CONTROL= on page 625</td>
</tr>
<tr>
<td>Specify whether to double space between the rows of the table</td>
<td>DOUBLE_SPACE on page 626</td>
</tr>
<tr>
<td>Specify whether extra space is evenly divided among all columns of the table</td>
<td>EVEN on page 626</td>
</tr>
<tr>
<td>Specify whether to split a long stacked column across page boundaries</td>
<td>SPLIT_STACK on page 630</td>
</tr>
<tr>
<td>Influence the display of the values in header cells and data cells</td>
<td></td>
</tr>
<tr>
<td>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</td>
<td>CLASSLEVELS= on page 625</td>
</tr>
<tr>
<td>Specify which format to use if both a column template and a data component specify a format</td>
<td>DATA_FORMAT_OVERRIDER on page 626</td>
</tr>
<tr>
<td>Specify whether to justify the format fields within the columns or to justify the values within the columns without regard to the format fields</td>
<td>JUSTIFY on page 627</td>
</tr>
<tr>
<td>Specify whether to order the columns by their order in the data component</td>
<td>ORDER_DATA on page 628</td>
</tr>
<tr>
<td>Specify the source of the values for the format width and the decimal width if they are not specified</td>
<td>USE_FORMAT_DEFAULTS on page 632</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Use the column name as the column header if neither the column</td>
<td>USE_NAME on page 632</td>
</tr>
<tr>
<td>template nor the data component specifies a header</td>
<td></td>
</tr>
<tr>
<td>Influence the layout of headers and footers</td>
<td></td>
</tr>
<tr>
<td>Specify the number of blank lines to place between the last row of</td>
<td>FOOTER_SPACE= on page 627</td>
</tr>
<tr>
<td>data and the first row of output</td>
<td></td>
</tr>
<tr>
<td>Specify the number of blank lines to place between the last row of</td>
<td>HEADER_SPACE= on page 627</td>
</tr>
<tr>
<td>headers and the first row of data</td>
<td></td>
</tr>
<tr>
<td>Specify whether to draw a continuous line above the first table</td>
<td>OVERLINE on page 628</td>
</tr>
<tr>
<td>footer or, if there is no table footer, below the last row of data</td>
<td></td>
</tr>
<tr>
<td>Specify whether to print table footers and any overlining of the</td>
<td>PRINT_FOOTERS on page 629</td>
</tr>
<tr>
<td>table footers</td>
<td></td>
</tr>
<tr>
<td>Specify whether to print table headers and any underlining of the</td>
<td>PRINT_HEADERS on page 629</td>
</tr>
<tr>
<td>table headers</td>
<td></td>
</tr>
<tr>
<td>Specify whether to draw a continuous line under the last table</td>
<td>UNDERLINE on page 631</td>
</tr>
<tr>
<td>header or, if there is no table header, then above the last row of</td>
<td></td>
</tr>
<tr>
<td>data on a page</td>
<td></td>
</tr>
<tr>
<td>Influence the non-LISTING output</td>
<td></td>
</tr>
<tr>
<td>Specify whether to place the output object in a table of contents, if</td>
<td>CONTENTS on page 625</td>
</tr>
<tr>
<td>you create a table of contents</td>
<td></td>
</tr>
<tr>
<td>Specify the label to use for the output object in the contents</td>
<td>CONTENTS_LABEL= on page 625</td>
</tr>
<tr>
<td>file, the Results window, and the trace record</td>
<td></td>
</tr>
<tr>
<td>Other table attributes</td>
<td></td>
</tr>
</tbody>
</table>
### Task
Control whether BY lines are printed above each BY group

### Attribute
BYLINE= on page 624

### Destinations
All except OUTPUT

### Task
Define the characters to use as the line-drawing characters in the table

### Attribute
FORMCHAR= on page 627

### Destinations
LISTING

### Task
Specify a label for the table

### Attribute
LABEL= on page 628

### Destinations
All

### Task
Specify the table that the current template inherits from

### Attribute
PARENT= on page 629

### Destinations
All

### Task
Specify the style element to use for the table and any changes to the attributes

### Attribute
"STYLE=<style-element-name>[style-attribute-specification(s)];" on page 630

### Destinations
Markup family, printer family, and RTF

### Task
Specify the special data set type of a SAS data set

### Attribute
"TYPE=string | variable;" on page 631

### Destinations
OUTPUT

---

**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.

Example: “Example 1: Creating a Stand-Alone Style” on page 459

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 16.7  Values in the Control Column

<table>
<thead>
<tr>
<th>Column Control Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A digit from 1-9</td>
<td>The specified number of blank lines precedes the current row.</td>
</tr>
<tr>
<td>A hyphen (-)</td>
<td>A row of underlining precedes the current row.</td>
</tr>
<tr>
<td>&quot;b&quot; or &quot;B&quot;</td>
<td>ODS tries to insert a panel break if the entire table does not fit in the space that is provided.</td>
</tr>
</tbody>
</table>

Default  None
Restriction  The "b" and "B" column control values are not supported for the PRINTER destination.
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.
Examples  “Example 1: Editing a Table Template That a SAS Procedure Uses” on page 551
          “Example 3: Creating a New Table Template ” on page 563

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.
**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.

Tips Use any character in formating-characters, including hexadecimal characters. If you use hexadecimal characters, then put an x after the closing quotation mark. For example, this option assigns the hexadecimal character 2-D to the first formatting character, the hexadecimal character 7C to the second character, and does not alter the remaining characters: formchar="2D7C"x

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

Interactions JUSTIFY=ON can interfere with decimal alignment.

- If the column is numeric, then values are aligned to the right if you specify JUSTIFY=OFF and JUST=C.
- All of the destinations except for the LISTING destination justify the values in columns as if JUSTIFY=ON for JUST=R and JUST=L.

Tips If you translate numeric data to character data, you might need to use JUSTIFY= to align the data.

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, or the first spanning header in the table.

Tip  The LABEL= attribute is valid in all destinations.

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.

Defaults  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

Tip  The OVERLINE attribute is valid only in the LISTING destination.

See  For information about formatting characters, see the discussion of "FORMCHAR= "string" | variable;" on page 627.

UNDERLINE= table attribute on page 631, UNDERLINE= column attribute on page 607, and the OVERLINE= column attribute on page 603.

Example  "Example 1: Editing a Table Template That a SAS Procedure Uses" on page 551
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 line size, display the data in multiple sets of
columns so that rows that would otherwise appear on multiple pages appear on
the same page.

Tips  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).

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

Tip  The PRINT_FOOTERS attribute is valid in all destinations except the
OUTPUT destination.

See  OVERLINE= table attribute on page 628

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. For
individual columns, use the following column attribute:
“PRINT_HEADERS<=ON | OFF | variable>,” on page 605.
Tip  The PRINT_HEADERS attribute is valid in all destinations except the OUTPUT destination.

See  “UNDERLINE<=ON | OFF | variable>;” on page 631

**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 448 ). 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  “Viewing the Contents of a Style” on page 429

“Finding and Viewing the Default Style for ODS Destinations” on page 429

For a table of style element names, see Chapter 20, “Style Elements,” on page 797.

**style-attribute-specification**
describes the style attribute to change. Each **style-attribute-specification** has this general form:

**style-attribute-name=style-attribute-value**

See  “About Style Attributes” on page 459

Default  Table
Requirement: Specify either a style-attribute-specification or a style-element-name with the STYLE= option.

Tips: You can use braces ({ and }) instead of square brackets ([ and ]). 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.

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 that a SAS data set type that you specify is a valid data set type or the structure of the data set that you create is appropriate for the type that you have assigned.

- **Tips**: 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.

  - 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

  - **Tip**: The UNDERLINE attribute is valid only in the LISTING destination.

  - **See**: For information about formatting characters, see "FORMCHAR= "string" | variable;" on page 627.

    - UNDERLINE= column attribute on page 607 and OVERLINE= column attribute on page 603.

    - Also see OVERLINE table attribute on page 628.

   - **Examples**: "Example 1: Editing a Table Template That a SAS Procedure Uses" on page 551

     - "Example 3: Creating a New Table Template " on page 563
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

Tips Use this attribute when column names are derived from a data set and 
the columns are generic.

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.

Tip The WRAP attribute is valid only in LISTING and printer family 
destinations.

See WRAP_SPACE table attribute on page 632 and ID= column 
attribute on page 601

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

Tip The WRAP_SPACE attribute is valid only in the LISTING, printer family, 
and RTF destinations.

See WRAP= table attribute on page 632
TEMPLATE Procedure: Creating Markup Language Tagsets

Overview: TEMPLATE Procedure: Creating Markup Language Tagsets ........ 633
Overview: ODS Tagsets and the TEMPLATE PROCEDURE .................. 634

Concepts: TEMPLATE Procedure: Creating Markup Language Tagsets .... 634
Getting Familiar with Tagsets ........................................... 634
Creating Custom Tagsets .............................................. 639

Syntax: TEMPLATE Procedure: Creating Markup Language Tagsets .... 642
DEFINE TAGSET Statement ........................................... 645
DEFINE EVENT Statement ........................................... 652
BLOCK Statement ..................................................... 655
BREAK Statement ...................................................... 655
CLOSE Statement ....................................................... 656
CONTINUE Statement ................................................ 656
DELSSTREAM Statement ............................................. 657
DO Statement .......................................................... 657
DONE Statement ........................................................ 658
ELSE Statement ........................................................ 658
END Statement .......................................................... 658
EVAL Statement ........................................................ 659
FLUSH Statement ...................................................... 661
ITERATE Statement ................................................... 661
NDENT Statement ........................................................ 662
NEXT Statement ....................................................... 663
NOTES Statement ...................................................... 664
OPEN Statement ........................................................ 664
PUT Statement .......................................................... 665
PUTL Statement ........................................................ 667
PUTLOG Statement ..................................................... 668
PUTQ Statement ....................................................... 670
PUTSTREAM Statement ............................................... 671
PUTVARS Statement .................................................. 672
SET Statement .......................................................... 674
STOP Statement ....................................................... 678
TRIGGER Statement ................................................... 679
UNBLOCK Statement .................................................. 680
UNSET Statement ...................................................... 680
XDENT Statement ...................................................... 682

Usage: TEMPLATE Procedure: Creating Markup Language Tagsets .... 683
Event Variables ......................................................... 683
Event Statement Conditions .......................................... 690

Examples: TEMPLATE Procedure: Creating Markup Language Tagsets .... 692
Overview: TEMPLATE Procedure: Creating Markup Language Tagsets

Overview: ODS Tagsets and the TEMPLATE PROCEDURE

The TEMPLATE procedure enables you to create a tagset. A tagset 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 “Using the TEMPLATE Procedure” on page 325.

Concepts: TEMPLATE Procedure: Creating Markup Language Tagsets

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:

```sas
proc template;
   list tagsets;
```
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” in SAS Output Delivery System: User’s Guide.

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 Tags to view the contents of that item store or directory.
  4. Double-click the tagset that you want 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**
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: Executing Events Using the TRIGGER= Statement” on page 705.

- Events can perform actions based on conditions.

- An event consists of PUT statements, text, and event variables.

For example, here is a simple event for an HTML table output:

```sas
define event table;
  start:
    put '<table>' nl;
  finish:
    put '</table>' nl;
end;
```

In the event:

1. The DEFINE EVENT statement begins the event and assigns it the name TABLE.
2. 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.

```sas
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 variables** are variables that include 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 variables** are variables that specify 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 about style attributes, see Chapter 14, “TEMPLATE Procedure,” on page 425.

- **dynamic variables** are variables that are dynamically created within SAS. Because these variables are dynamically created, their names, or how they are used, are 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 variables** are arrays that contain 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 variables**

are arrays that contain 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 variables**

are variables that are 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. For more information, see the "MVAR Statement" on page 539 and "NMVAR Statement" on page 540.

**Memory variables**

are 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 variables**

are areas of memory that contain 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 variables**

are temporary item stores that contain 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:

```sas
define event colsspecs;
   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 639. For a list of event variables and their descriptions, see “Event Variables” on page 683.

---

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.Csv all t inherits most of its events from Tagsets.Csv, and Tagsets.MSOffice2k gets most of its events from Tagsets.Html4. 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: Creating a Tagset through Inheritance” on page 692.

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:

```sas
ods listing;
ods markup type=event_map file='custom-tagset-filename.xml';
proc print data=sashelp.class;
  where Height gt 60;
run;
ods markup close;
ods listing close;
```

Here is the listing output and resulting XML file:

**Output 17.1 LISTING Output**

![Output 17.1 LISTING Output](image-url)
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 reflects only the events that you defined.
Note: When you first run a SAS process and specify TYPE=Event_Map, you can also generate a style sheet along with the body file. The style sheet 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:

- The Text_Map tagset generates output that is similar to a LISTING output.
- The Style_Popup tagset generates HTML like HTMLCSS. However, in Internet Explorer, Style_Popup displays a window that shows the resolved ODS style template for any item that you click.
- The Style_Display tagset is similar to the Style_Popup tagset, but it generates a simple page of output for you to click.
- The NamedHtml tagset generates HTML output similar to the Style_Popup tagset 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 about DATA step functions and statements, see SAS Functions and CALL Routines: Reference, SAS DATA Step Statements: Reference, and SAS Language Reference: Concepts.

Syntax: TEMPLATE Procedure: Creating Markup Language Tagsets

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;> >
BLOCK event-name < / event-statement-condition(s)>;
BREAK < / event-statement-condition(s)>;
CLOSE < / event-statement-condition(s)>;
CONTINUE < / event-statement-condition(s)>;
DELSSTREAM $$stream-variable-name < / event-statement-condition(s)>;
DO < / event-statement-condition(s)>;
DONE;
ELSE </event-statement-condition(s)>;
EVAL $$\langle user-defined-variable \text{ where-expression}\rangle$$ < / event-statement-condition(s)>;
FLUSH <event-statement-condition(s)>;
ITERATE $$\langle dictionary-variable \mid list-variable\rangle$$ < / event-statement-condition(s)>;
NEXT $$\langle dictionary-variable \mid list-variable\rangle$$ < / event-statement-condition(s)>;
OPEN $$\langle stream-variable-name\rangle$$ < / event-statement-condition(s)>;
PUT <function> <NL> <variable> <text> < / event-statement-condition(s)>;
PUTL ( <variable> <text> | <function> | <NL>) < / event-statement-condition(s)>;
PUTLOG ( <variable> <text> <function>) < / event-statement-condition(s)>;
PUTQ ( <variable> <text> | <function> | <NL>) < / event-statement-condition(s)>;
PUTSTREAM $$\langle stream-variable-name\rangle$$ < / event-statement-condition(s)>;
PUTVARS variable-group variable-group-value < / event-statement-condition(s)>;
SET $$\langle user-defined-variable-name \text{ user-defined-variable-value}\rangle$$ < / event-statement-condition(s)>;
STOP < / event-statement-condition(s)>;
TRIGGER event-name <START | FINISH> < / event-statement-condition(s)>;
UNBLOCK event-name < / event-statement-condition(s)>;
UNSET ALL | $$\langle memory-variable\rangle$$ | $$\langle stream-variable\rangle$$ < / event-statement-condition(s)>;
XDENT < / event-statement-condition(s)>;
END;
NOTES;
END;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE TAGSET</td>
<td>Create a tagset</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 8</td>
</tr>
<tr>
<td>DEFINE EVENT</td>
<td>Define what is written to the output file</td>
<td>Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7</td>
</tr>
<tr>
<td>BLOCK</td>
<td>Disable an event</td>
<td></td>
</tr>
<tr>
<td>BREAK</td>
<td>Stop the current event from executing</td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>Close the current stream variable and direct all future output to the output file</td>
<td></td>
</tr>
<tr>
<td>CONTINUE</td>
<td>Return the execution of the DO loop to the corresponding DO statement for re-evaluation of the IF event statement condition</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Task</td>
<td>Example</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>DELSTREAM</td>
<td>Delete the specified stream variable</td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>Begin a statement block that executes if the required condition is true</td>
<td></td>
</tr>
<tr>
<td>DONE</td>
<td>End a DO or ELSE statement block</td>
<td></td>
</tr>
<tr>
<td>ELSE</td>
<td>Begin a statement block that executes if the corresponding DO statement is false</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Specify one or more conditions that must be true for a DEFINE EVENT statement to execute</td>
<td></td>
</tr>
<tr>
<td>&quot;Event Variables&quot;</td>
<td>Set one or more event attributes</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>End the event</td>
<td></td>
</tr>
<tr>
<td>EVAL</td>
<td>Create or update a user-defined variable</td>
<td></td>
</tr>
<tr>
<td>FLUSH</td>
<td>Write buffered output to the current output file or the current stream variable</td>
<td></td>
</tr>
<tr>
<td>ITERATE</td>
<td>Iterate through a dictionary variable or list variable and assign its value to the <em>NAME</em> and <em>VALUE</em> event variables for each iteration</td>
<td></td>
</tr>
<tr>
<td>NDENT</td>
<td>Indent output one more level than specified by the INDENT= attribute</td>
<td>Ex. 3, Ex. 5</td>
</tr>
<tr>
<td>NEXT</td>
<td>Increase a dictionary or list variable incrementally to the next value and repopulate event variables</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>Provide information about the tagset</td>
<td>Ex. 3</td>
</tr>
<tr>
<td>OPEN</td>
<td>Open or create a stream variable</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td>Write text, new lines, variable values, or DATA step function return values to an output file</td>
<td>Ex. 1, Ex. 3, Ex. 4, Ex. 5, Ex. 6</td>
</tr>
<tr>
<td>PUTL</td>
<td>Write text, new lines, variable values, or DATA step function return values to an output file and add a new line to the end of the output</td>
<td></td>
</tr>
<tr>
<td>PUTLOG</td>
<td>Write text, new lines, variable values, or DATA step function return values to the log</td>
<td></td>
</tr>
<tr>
<td>PUTQ</td>
<td>Write text, new lines, variable values, or DATA step function return values to an output file and enclose in quotation marks</td>
<td>Ex. 7</td>
</tr>
<tr>
<td>PUTSTREAM</td>
<td>Write the contents of the specified stream variable to an output file</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Task</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>PUTVARS</td>
<td>Write text, new lines, variable values, or DATA step function return values to an output file for each value in a variable group, list, or dictionary</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td>Create or update a user-defined variable and its value</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>Move the execution to the end of the current statement block</td>
<td></td>
</tr>
<tr>
<td>TRIGGER</td>
<td>Execute an event</td>
<td>Ex. 3, Ex. 4, Ex. 5, Ex. 6</td>
</tr>
<tr>
<td>UNBLOCK</td>
<td>Enable a disabled event</td>
<td></td>
</tr>
<tr>
<td>UNSET</td>
<td>Delete a user-defined variable and its value</td>
<td></td>
</tr>
<tr>
<td>XDENT</td>
<td>Indent output one less indentation level</td>
<td>Ex. 3, Ex. 5</td>
</tr>
</tbody>
</table>

**DEFINE TAGSET Statement**

Creates a tagset.

**Requirement:** An END statement must be the last statement in the template.

**Syntax**

```
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;
```

**Summary of Optional Arguments**

`STORE=libref.template-store access-options`

**Required Arguments**

`tagset-path`

specifies where to store the tagset.

**Default** PROC TEMPLATE writes the template to the first template store in the current path where you have Write access.
A tagset-path consists of one or more names that are separated by periods. Each name represents a directory, or level, in a template store.

Tips

Use the ODS PATH statement to control the item store where the tagset is stored.

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. 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.Tagset supplied by SAS contains information used by many tagsets. If this information is not retained, unexpected behavior might 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 635.

**Optional Argument**

**STORE=libref.template-store access-options**
specifies the template store where the template is stored.

**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.

**access-options**
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 replaces it.
Restriction  
The STORE= option syntax does not become part of the compiled template.

Interaction  
The STORE= option overrides the search list specified in the PATH statement.

Tagset Attributes

Table 17.1  Tagset Attributes by Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the maximum number of characters that will be considered for forced line breaks by ODS</td>
<td>BREAKTEXT_LENGTH= (p. 648)</td>
</tr>
<tr>
<td>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</td>
<td>BREAKTEXT_RATIO= (p. 648)</td>
</tr>
<tr>
<td>Specify the maximum width of space available for text entry that ODS will consider for placement of automatic line breaks</td>
<td>BREAKTEXT_WIDTH= (p. 649)</td>
</tr>
<tr>
<td>Specify the text to use as a copyright</td>
<td>COPYRIGHT= (p. 649)</td>
</tr>
<tr>
<td>Specify the name of the event to use by default</td>
<td>DEFAULT_EVENT= (p. 649)</td>
</tr>
<tr>
<td>Specify whether the tagset supports embedded style sheets</td>
<td>EMBEDDED_STYLESHEET= (p. 649)</td>
</tr>
<tr>
<td>Specify a comma-delimited list of image types or file extensions that are valid for an output destination</td>
<td>IMAGE_FORMATS= (p. 649)</td>
</tr>
<tr>
<td>Specify the number of spaces the NDENT and XDENT event statements indent the output</td>
<td>INDENT= (p. 650)</td>
</tr>
<tr>
<td>Specify a string, which is printed to the SAS log when the tagset is used</td>
<td>LOG_NOTE= (p. 650)</td>
</tr>
<tr>
<td>Specify special characters and their translations</td>
<td>MAP= (p. 650)</td>
</tr>
<tr>
<td>Specify strings to substitute for special characters</td>
<td>MAPSUB= (p. 650)</td>
</tr>
<tr>
<td>Set a category for the output</td>
<td>OUTPUT_TYPE= (p. 651)</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify whether a byte-order mark is written to the output files when</td>
<td>NO_BYTE_ORDER_MARK= (p. 651)</td>
</tr>
<tr>
<td>using a UTF character set</td>
<td></td>
</tr>
<tr>
<td>Define a nonbreaking space for the markup output</td>
<td>NOBREAKSPACE= (p. 650)</td>
</tr>
<tr>
<td>Specify the tagset from which the current template inherits</td>
<td>PARENT= (p. 651)</td>
</tr>
<tr>
<td>Specify whether all style attributes are available at all times</td>
<td>PURE_STYLE= (p. 651)</td>
</tr>
<tr>
<td>Specify the text to use as a registered trademark</td>
<td>REGISTERED_TM= (p. 652)</td>
</tr>
<tr>
<td>Define a string to use for line breaks in the markup output</td>
<td>SPLIT= (p. 652)</td>
</tr>
<tr>
<td>Specify whether the tagset lets procedures place columns on top of</td>
<td>STACKED_COLUMNS= (p. 652)</td>
</tr>
<tr>
<td>each other, or side by side</td>
<td></td>
</tr>
<tr>
<td>Specify the text to use as a trademark</td>
<td>TRADEMARK= (p. 652)</td>
</tr>
</tbody>
</table>

**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 on the total available text width.

Example

To instruct ODS not to 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 than 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 that 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.

Example  "Example 3: Creating a New Tagset" on page 701

EMBEDDED_STYLESHEET= YES | NO

specifies whether the tagset supports embedded style sheets.

YES

supports embedded style sheets.

Alias  ON

NO

does not support embedded style sheets.

Alias  OFF

Default  YES

Tip  If embedded style sheets are supported and you do not specify a style sheet in the ODS statement, then the style sheet is written to the top of the output file.

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 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.

Examples “Example 3: Creating a New Tagset” on page 701
“Example 5: Indenting Output” on page 707

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.

Requirements When listing special characters in the MAP= statement, omit blank spaces between them.
When you specify special characters, enclose the list of special characters in quotation marks.
Use the MAP= statement with the MAPSUB statement.

Example “Example 3: Creating a New Tagset” on page 701

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.

Requirements When specifying multiple strings, use a forward slash (/) to separate the text strings.
When specifying strings, enclose the entire string list in quotation marks.
Use the MAPSUB= statement with the MAP= statement.

Example “Example 3: Creating a New Tagset” on page 701

NOBREAKSPACE= 'string'
defines a nonbreaking space for the markup output.
**string**
specifies the character that defines a nonbreaking space.

**Restriction**
Specify only one string at a time.

**Requirement**
When specifying a string, enclose the string in quotation marks.

**Example**
“Example 3: Creating a New Tagset” on page 701

**NO_BYTE_ORDER_MARK=YES | ON | NO | OFF**
specifies whether 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.

**Example**
“Example 3: Creating a New Tagset” on page 701

**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.

  **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.

  **Tips**
Specify a tagset that SAS supplies or a customized tagset.

  **Examples**
“Example 1: Creating a Tagset through Inheritance” on page 692

  “Example 8: Using the STACKED_COLUMNS Attribute in a Tagset” on page 711

**PURE_STYLE=YES | NO**
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.

Restriction Specify one string at a time.

Requirement When specifying a string, enclose the string in quotation marks.

Example “Example 3: Creating a New Tagset” on page 701

STACKED_COLUMNS= YES | NO
specifies whether the tagset lets procedures stack columns on top of each other, or place them side by side.

YES
stacks columns on top of each other.

Alias ON

NO
stacks columns side by side.

Alias OFF

Default YES.

Tip To place columns side by side, specify the NO or OFF value.

Examples “Example 3: Creating a New Tagset” on page 701

“Example 8: Using the STACKED_COLUMNS Attribute in a Tagset” on page 711

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 690.

Examples: “Example 6: Using Different Styles for Events” on page 709

“Example 7: Modifying an Event to Include Other Style Sheets” on page 711
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 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)>;
UNSET ALL | $$memory-variable | $$stream-variable< / event-statement-condition(s)>;
XDENT < / event-statement-condition(s)>;
END;

Summary of Optional Arguments

FILE= BODY | CODE | CONTENTS | DATA | FRAME | PAGES | STYLESHEET;
Redirect event output to any of the known types of output that are open
PURE_STYLE= YES | NO;
Enable the event to use any style element that has been defined
STYLE= style-element;
Specify a style element
Required Argument

*event-name*

specifies the name of the event.

Event Attributes

**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 filenames in 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” in *SAS Output Delivery System: User's Guide*.

**See**


**PURE_STYLE= YES | NO;**

specifies whether to enable the event to use any of the style elements that have been defined.

**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

**Default**

NO

**Restriction**

The PURE_STYLE= attribute is valid only in markup family destinations.

**See**

“DEFINE STYLE Statement” on page 448

**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**

“DEFINE STYLE Statement” on page 448

**Example**

“Example 6: Using Different Styles for Events” on page 709
BLOCK Statement
Disables the specified event.

Tips:
- To enable the blocked event, use the UNBLOCK statement.
- You can block the same event multiple times, but to enable the event, use the same number of UNBLOCK statements.

Syntax
```
BLOCK event < / event-statement-condition(s)>;
```

Required Argument
```
event
```
- specifies the event.

Optional Argument
```
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
For information about these conditions, see “Event Statement Conditions” on page 690.

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.

Syntax
```
BREAK < / event-statement-condition(s)>;
```

Optional Argument
```
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 (/).
CLOSE Statement
Closes the current stream variable and directs all future output to the output file.

Syntax
CLOSE < / event-statement-condition(s)>;

Optional Argument

**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
For information about these conditions, see “Event Statement Conditions” on page 690.

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:
“DO Statement” on page 657

Syntax
CONTINUE < / event-statement-condition(s)>;

Optional Argument

**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
For information about these conditions, see "Event Statement Conditions" on page 690.
DELSTREAM Statement
Deletes the specified stream variable.

Syntax
DELSTREAM stream-variable < /event-statement-condition(s)>;

Required Argument
stream-variable
specifies the stream variable to be deleted.

See "OPEN Statement" on page 664

Optional Argument
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 For information about these conditions, see "Event Statement Conditions" on page 690.

DO Statement
Begins a statement block that executes if the required condition is true.

Syntax
DO /event-statement-condition(s);

Required Argument
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 For information about these conditions, see "Event Statement Conditions" on page 690.
DONE Statement

Ends a DO or ELSE statement block.

See:

“DO Statement” on page 657
“ELSE Statement” on page 658

Syntax

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: “DO Statement” on page 657

Syntax

ELSE < / event-statement-condition(s)>;

Optional Argument

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 For information about these conditions, see “Event Statement Conditions” on page 690.

END Statement

Ends the tagset or event.

Syntax

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.

Syntax

**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 that 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 delete them with the UNSET Statement on page 680.

**See** For more information, see “Understanding Variables” on page 637.

**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.

Requirements

Specify brackets [], even if you omit an index.

Enclose index in brackets.

See

For more information, see “Understanding Variables” on page 637.

Requirement

list-variable must be preceded by a '$' symbol.

Tips

List variables are accessed sequentially by using the “ITERATE Statement” on page 661 and the “NEXT Statement” on page 663.

After you create list variables, they are globally available in all events until you use the “UNSET Statement” on page 680 to delete them.

scalar variable

specifies a scalar variable to which to assign a where-expression return value.

Requirements

Scalar variables must be preceded by the '$' symbol.

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

For information about stream variables, see “Understanding Variables” on page 637.

where-expression

any expression that can be used in the WHERE= data set option.

See

For information about expressions that you can use in the WHERE data set option, see the “WHERE= Data Set Option” in SAS Data Set Options: Reference. Also see “WHERE-Expression Processing” in SAS Language Reference: Concepts.

Optional Argument

event-statement-condition(s)

specifies one or more conditions that must be true for the event statement to execute.
**FLUSH Statement**

Writes buffered output to the current output file or the current stream variable.

**Syntax**

```
FLUSH <\ event-statement-condition(s)\>; 
```

**Optional Argument**

`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**

For information about these conditions, see “Event Statement Conditions” on page 690.

---

**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:**

`_VALUE_` and `_NAME_` in Table 17.65 on page 683.

**Syntax**

```
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.
list-variable

specifies a list variable.

Requirement list-variable must be preceded by the "$" symbol.

Tip User-defined variables are not case sensitive.

See The “EVAL Statement” on page 659 or the “SET Statement” on page 674 for information about list variables

Optional Argument

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 For information about these conditions, see “Event Statement Conditions” on page 690.

NDENT Statement

Indents output one more level than the number of spaces specified by the INDENT= attribute.

Interaction: The start position of the indentation level is set by the INDENT= attribute.

Examples: “Example 3: Creating a New Tagset” on page 701
“Example 5: Indenting Output” on page 707

Syntax

NDENT < / event-statement-condition(s)>;

Optional Argument

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 For information about these conditions, see “Event Statement Conditions” on page 690.
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: _VALUE_ and _NAME_ in Table 17.65 on page 683.

Syntax

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 "ITERATE Statement" on page 661

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 "ITERATE Statement" on page 661

Optional Argument

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 For information about these conditions, see "Event Statement Conditions" on page 690.
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.

**See:**
- “Example 3: Creating a New Tagset” on page 701
- “Example 8: Using the STACKED_COLUMNS Attribute in a Tagset” on page 711

**Syntax**

```plaintext
NOTES 'text';
```

**Required Argument**

`text`

provides information about the tagset.

**Requirement** When specifying `text`, enclose the text in quotation marks.

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.

**Syntax**

```plaintext
OPEN stream-variable <! event-statement-condition(s)>;
```

**Required Argument**

`stream-variable`

specifies a stream variable, which is a temporary item store that contains output.

**Tips** User-defined variables are not case sensitive.

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:

```plaintext
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is 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 “memory variables” on page 638.

Optional Argument

event-statement-condition(s)

specifies one or more conditions that must be true for the event statement to execute.

Requirement \textit{event-statement-condition(s)} must be preceded by a slash (/).

See For information about these conditions “Event Statement Conditions” on page 690.

PUT Statement

Writes text, new lines, variable values, or DATA step function return values to an output file.

Examples:

“Example 1: Creating a Tagset through Inheritance” on page 692
“Example 3: Creating a New Tagset” on page 701
“Example 4: Executing Events Using the TRIGGER= Statement” on page 705
“Example 5: Indenting Output” on page 707
“Example 6: Using Different Styles for Events” on page 709

Syntax

\texttt{PUT <text> <NL(s)> <value(s)> \textit{<event-statement-condition(s)>};

Optional Arguments

\texttt{NL}

specifies a new line.

\texttt{text}

specifies a string of text.

\texttt{text} must be enclosed in quotation marks.

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.

**Interactions**
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**
For information about DATA step functions, see [SAS Functions and CALL Routines: Reference](#).

For information about variables, see "Understanding Variables" on page 637.

"Event Variables" on page 683 for a list of event variables

**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**
For information about these conditions, see “Event Statement Conditions” on page 690.
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.

Syntax

```
PUTL text NL value(s) / event-statement-condition(s);
```

Optional Arguments

- **NL**
  - Specifies a new line.
  - Aliases: CR, 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.
  - Interactions: 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:

```
putl '<table' 'background=' background 'foreground=' foreground
  'cellpadding=' cellpadding '>' nl;
```
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
For information about DATA step functions, see SAS Functions and CALL Routines: Reference.

For information about variables, see “Understanding Variables” on page 637.

“Event Variables” on page 683 for a list of event variables

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
For information about these conditions, see “Event Statement Conditions” on page 690.

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.

Syntax

PUTLOG <text> <value(s)> </ event-statement-condition(s)>;

Optional 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
For information about these conditions, see “Event Statement Conditions” on page 690.
text
specifies a string of text.

Requirement  text must be enclosed in quotation marks.

Interactions  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 '>';
PUTQ Statement

Writes text, new lines, variable values, or DATA step function return values to an output file and places quotation marks around the value of the variable.

Example:  

“Example 7: Modifying an Event to Include Other Style Sheets” on page 711

Syntax

PUTQ <text> <NL(s)> <value(s)> </event-statement-condition(s)>;

Optional Arguments

NL

specifies a new line.

Aliases  

CR

LF

text

specifies a string of text.

Requirement  text must be enclosed in quotation marks.

Interactions 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.

Interactions 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
For information about DATA step functions, see SAS Functions and CALL Routines: Reference.
For information about variables, see “Understanding Variables” on page 637.
“Event Variables” on page 683 for a list of event variables

**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
For information about these conditions, see “Event Statement Conditions” on page 690.

---

## PUTSTREAM Statement

Writes the contents of the specified stream variable to an output file.

### Syntax

**PUTSTREAM** stream-variable < / event-statement-condition(s)>;

**Required Argument**

**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:

```sas
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is used as a stream variable';
```
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 “memory variables” on page 638.

Optional Argument

**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  
For information about these conditions, see “Event Statement Conditions” on page 690.

---

**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:  
_VALUE_ and __NAME__ in Table 17.65 on page 683.

---

**Syntax**

```
PUTVARS (variable-group | dictionary-variable | list-variable)<NL(s)> <'text'>
<value(s)>  
< / event-statement-condition(s)>;
```

**Required Arguments**

**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  
“Event Variables” on page 683
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 PUTVARS 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  For information about list variables, see the following sections: "EVAL Statement" on page 659, "SET Statement" on page 674, and "Understanding Variables" on page 637.

list-variable
specifies a list variable.

Requirement  list-variable must be preceded by the "$" symbol.

Tip  User-defined variables are not case sensitive.

See  For information about list variables, see the following sections: "EVAL Statement" on page 659, "SET Statement" on page 674, and "Understanding Variables" on page 637.

Optional Arguments

NL
specifies a new line.

Aliases  CR
         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  For information about DATA step functions, see *SAS Functions and CALL Routines: Reference*.

For information about variables, see “Understanding Variables” on page 637.

For a list of event variables, see “Event Variables” on page 683.

**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  For information about these conditions, see “Event Statement Conditions” on page 690.

---

**SET Statement**

Creates or updates a user-defined variable and its value.

**Syntax**

\[
\text{SET } \$\text{dictionary-variable entry} < /\text{event-statement-condition(s)}> ; \\
\text{SET } \$\text{list-variable entry} < /\text{event-statement-condition(s)}> ; \\
\text{SET } \$\text{scalar-variable} | \$\$\text{stream-variable entry} < /\text{event-statement-condition(s)}> ;
\]

**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:

\[
\$\text{dictionary-variable}['\text{key'}]
\]

['*key*']
specifies a subscript that contains text or a variable that has a character value.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Enclose key in quotation marks and brackets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>key is case preserving and case sensitive.</td>
</tr>
</tbody>
</table>
| 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. |
| Tips | Dictionary variables are accessed sequentially by using the ITERATE and NEXT statements. See “ITERATE Statement” on page 661. Also see “NEXT Statement” on page 663. After they are created, dictionary variables are globally available in all events until you delete them by using the “UNSET Statement” on page 680. |

**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 | [SAS Functions and CALL Routines: Reference](#) for information about 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.
  - Tips | 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 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;

User-defined variables are not case sensitive.

See "memory variables" on page 638

For information about variables, see "Understanding Variables" on page 637.

"Event Variables" on page 683 for a list of event variables

**list-variable**
specifies an array that contains a list of numbers or strings of text that are indexed.

**list-variable** has the following form:

\$list-variable[<index>]\n
[<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.

Requirements Specify brackets [], even if you omit an index.

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.

Tips List variables are accessed sequentially by using the **ITERATE** and **NEXT** statements. See "ITERATE Statement" on page 661 and "NEXT Statement" on page 663.

After they are created, list variables are globally available in all events until you delete them using the "UNSET Statement" on page 680.
**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" on page 680.

**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:

```plaintext
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is used as a stream variable';
close;
```

Therefore, these statements are equivalent:

```plaintext
put $$test;
putstream $mystream;
putstream test;
```

These statements are also equivalent:

```plaintext
unset $$test;
delstream $mystream;
delstream test;
```

**See**
"variable" on page 675
"memory variables" on page 638

---

**Optional Argument**

**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**
For information about these conditions, see "Event Statement Conditions" on page 690.

---

**Adding Entries to Dictionary Variables**

Use this form of the SET statement to add an entry to a dictionary variable.

```plaintext
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:

```plaintext
set $mydictionary['URL1'] 'links internally';
set $mydictionary['URL2'] 'links externally';
```

Adding Entries to List Variables

Use this form of the SET statement to add an entry to a list variable.

```plaintext
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.

**Syntax**

```plaintext
STOP */event-statement-condition(s)>;
```

**Optional Argument**

`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**

For information about these conditions, see “Event Statement Conditions” on page 690.
TRIGGER Statement

Executes an event.

Tip: The TRIGGER statement explicitly requests a specific action of an event.

Examples:
- “Example 3: Creating a New Tagset” on page 701
- “Example 4: Executing Events Using the TRIGGER= Statement” on page 705
- “Example 5: Indenting Output” on page 707
- “Example 6: Using Different Styles for Events” on page 709

Syntax

```
TRIGGER event-name <START | FINISH> <\ event-statement-condition(s)>;
```

Without Arguments

If a triggered event does not have start or finish sections, then it runs the current event statements.

Required Argument

`event-name`

specifies the name of the event.

Optional Arguments

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

For information about these conditions, see “Event Statement Conditions” on page 690.
UNBLOCK Statement

Enables a disabled event.

Requirement: Because you can block the same event multiple times, to enable the event use the same number of UNBLOCK statements as BLOCK statements.

Interaction: To disable an event, use the BLOCK statement.

Syntax

UNBLOCK event-name event-statement-condition(s);

Required Argument

event-name
specifies the name of the event.

Optional Argument

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 For information about these conditions, see "Event Statement Conditions" on page 690.

UNSET Statement

Deletes a user-defined variable and its value.

Syntax

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.

Requirements

- Enclose key in quotation marks and brackets.

- 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.

Requirements

- Specify brackets [], even if you omit an index.

- index must be number or numeric variable.

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

“SET Statement” on page 674 or “Understanding Variables” on page 637 for information about 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 \textit{stream-variable}, then the stream variable resolves as the value of the memory variable. For example, the following program uses the memory variable \texttt{$MyStream} as a stream variable:

\begin{verbatim}
set $mystream 'test';
open $mystream;
put 'The memory variable $mystream is used as a stream variable';
close;
\end{verbatim}

Therefore, the following statements are equivalent:

\begin{verbatim}
put $$test;
putstream $mystream;
putstream test;
\end{verbatim}

The following statements are also equivalent:

\begin{verbatim}
unset $$test;
delstream $mystream;
delstream test;
\end{verbatim}

See “SET Statement” on page 674 or “Understanding Variables” on page 637 for information about memory variables.

Optional Argument

event-statement-condition(s)  
specifies one or more conditions that must be true for the event statement to execute.

Requirement  
An \textit{event-statement-condition} must be preceded by a slash (/).

See  
For information about these conditions, see “Event Statement Conditions” on page 690.

XDENT Statement

Indents output one less indentation level, using the number of spaces specified by the \texttt{INDENT=} attribute.

Interaction:  
The starting level of indentation is set by the \texttt{NDENT=} statement.

Examples:  
“Example 3: Creating a New Tagset” on page 701
“Example 5: Indenting Output” on page 707

Syntax

\begin{verbatim}
XDENT < / event-statement-condition(s) > ;
\end{verbatim}

Optional Argument

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: For information about these conditions, see “Event Statement Conditions” on page 690.

---

Usage: TEMPLATE Procedure: Creating Markup Language Tagsets

### 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.

SAS includes these accessibility and compatibility features to improve the usability of SAS for users with a wide range of abilities.

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBR</td>
<td>Specifies an abbreviation for the event variable.</td>
</tr>
<tr>
<td>ACRONYM</td>
<td>Specifies an acronym for an event variable.</td>
</tr>
<tr>
<td>ALT</td>
<td>Specifies an alternate description of an event variable.</td>
</tr>
<tr>
<td>CAPTION</td>
<td>Specifies the caption for a table.</td>
</tr>
<tr>
<td>LONGDESC</td>
<td>Specifies the long description of an event variable.</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>Specifies a summary of a table.</td>
</tr>
</tbody>
</table>

### Table 17.2 Accessibility Variables

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>NAME</em></td>
<td>Contains the name of the current variable.</td>
</tr>
<tr>
<td>Event Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>VALUE</td>
<td>Contains the value of the current variable.</td>
</tr>
<tr>
<td>DNAME</td>
<td>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 &quot;DATANAME=column-name;&quot; on page 598.</td>
</tr>
<tr>
<td>LABEL</td>
<td>Specifies a label for the variable. The LABEL event variable is set with the LABEL= attribute in the column template. For information, see the &quot;LABEL=&quot;text&quot;</td>
</tr>
<tr>
<td>NAME</td>
<td>Specifies the name of the variable. NAME is set with the VARNAME= attribute in the column template. For information, see the &quot;VARNAME=variable-name</td>
</tr>
<tr>
<td>VALUE</td>
<td>Specifies the current value.</td>
</tr>
<tr>
<td>VALUECOUNT</td>
<td>Specifies the count of the variable.</td>
</tr>
</tbody>
</table>

**Table 17.4 Event Meta Variables**

<table>
<thead>
<tr>
<th>Event Meta Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPTY</td>
<td>Sets a flag to determine whether an event is called as an empty tag.</td>
</tr>
<tr>
<td>EVENT_NAME</td>
<td>Specifies the requested event name.</td>
</tr>
<tr>
<td>STATE</td>
<td>Specifies the current state of the event, which is either START or FINISH.</td>
</tr>
<tr>
<td>TRIGGER_NAME</td>
<td>Specifies the name of the event that is triggered.</td>
</tr>
</tbody>
</table>

**Table 17.5 Formatting Data**

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSURE</td>
<td>Specifies whether the endpoints of a format range are included or excluded, for example (&lt;-, -, &lt;-, &lt;=, and so on).</td>
</tr>
<tr>
<td>Event Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COL_ID</td>
<td>Specifies the column ID to identify columns. Used for the OIMDBM format type by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>DATAENCODING</td>
<td>Specifies the encoding type for Raw value. It is always Base64.</td>
</tr>
<tr>
<td>MISSING</td>
<td>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.</td>
</tr>
<tr>
<td>NO_WRAP</td>
<td>Specifies that the current cell should not wrap text or insert hyphens.</td>
</tr>
<tr>
<td>PRECISION</td>
<td>Specifies the number of places to the right of the decimal. The PRECISION variable is used by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>RANGEEND</td>
<td>Specifies the end value of a range in a format.</td>
</tr>
<tr>
<td>RANGESTART</td>
<td>Specifies the start value of a range in a format</td>
</tr>
<tr>
<td>RAWVALUE</td>
<td>Specifies the base64 encoding of the stored machine representation of the original value.</td>
</tr>
<tr>
<td>SASFORMAT</td>
<td>Specifies the SAS format used to format a value.</td>
</tr>
<tr>
<td>SCALE</td>
<td>Specifies the total number of places in the floating point number. The SCALE event variable is used by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Specifies the STRING, DOUBLE, CHAR, BOOL, or INT data type.</td>
</tr>
<tr>
<td>UNFORMATTEDTYPE</td>
<td>Specifies the data type before formatting.</td>
</tr>
<tr>
<td>UNFORMATTEDVALUE</td>
<td>Specifies the value before formatting.</td>
</tr>
<tr>
<td>UNFORMATTEDWIDTH</td>
<td>Specifies the width before formatting.</td>
</tr>
</tbody>
</table>
### Table 17.6  General Use Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANCHOR</td>
<td>Specifies the current anchor, which is the last value of the anchor tag (for example, IDX).</td>
</tr>
<tr>
<td>DATA_VIEWER</td>
<td>Specifies the name of the Data Viewer, such as Table, Batch, Tree, Graph, Report, or Print.</td>
</tr>
<tr>
<td>DATE</td>
<td>Specifies the date.</td>
</tr>
<tr>
<td>DEST_FILE</td>
<td>Specifies the current destination file, which is one of the following: body, contents, pages, frame, code, or style sheet.</td>
</tr>
<tr>
<td>FIRSTPAGE</td>
<td>Specifies the first page of the output file.</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>Specifies the language of the current output. The LANGUAGE event variable is set only when it is an Asian language.</td>
</tr>
<tr>
<td>OUTPUT_LABEL</td>
<td>Specifies the label of the current output object.</td>
</tr>
<tr>
<td>OUTPUT_NAME</td>
<td>Specifies the name of the current output object.</td>
</tr>
<tr>
<td>OUTPUT_TYPE</td>
<td>Specifies the output type as specified in the tagset.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>Specifies the page count since the files were opened.</td>
</tr>
<tr>
<td>PROC_COUNT</td>
<td>Specifies how many procedures have run since the files were opened.</td>
</tr>
<tr>
<td>PROC_NAME</td>
<td>Specifies the name of the current procedure.</td>
</tr>
<tr>
<td>SASLONGVERSION</td>
<td>Specifies the long format of the SAS version.</td>
</tr>
<tr>
<td>SASVERSION</td>
<td>Specifies the short format of the SAS version.</td>
</tr>
<tr>
<td>SPACE</td>
<td>Specifies the string that the tagset uses for a nonbreaking space.</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Specifies the string that the tagset uses for line breaks.</td>
</tr>
<tr>
<td>STYLE</td>
<td>Specifies the current style that is in use.</td>
</tr>
<tr>
<td>STYLE_ELEMENT</td>
<td>Specifies the name of the current style element.</td>
</tr>
<tr>
<td>SUPPRESS_CHARSET</td>
<td>Specifies the Suppress Charset Registry setting.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>TIME</td>
<td>Specifies the time.</td>
</tr>
<tr>
<td>TOCLEVEL</td>
<td>Specifies the table of contents level.</td>
</tr>
<tr>
<td>TOTAL_PAGE_COUNT</td>
<td>Specifies the total page count since ODS was opened.</td>
</tr>
<tr>
<td>TOTAL_PROC_COUNT</td>
<td>Specifies the number of procedures that have run since ODS was opened.</td>
</tr>
</tbody>
</table>

**Table 17.7  ODS Statement Variables: Variables That Originate with the ODS Statement That Invoked the Tagset**

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOR</td>
<td>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.</td>
</tr>
<tr>
<td>BASENAME</td>
<td>Specifies the name of the BASE= option as set in an ODS statement.</td>
</tr>
<tr>
<td>BODY_NAME</td>
<td>Specifies the name of the body file.</td>
</tr>
<tr>
<td>BODY_TITLE</td>
<td>Specifies the title of the body file.</td>
</tr>
<tr>
<td>BODY_URL</td>
<td>Specifies the URL of the body file.</td>
</tr>
<tr>
<td>CODE_NAME</td>
<td>Specifies the name of the code file.</td>
</tr>
<tr>
<td>CODE_TITLE</td>
<td>Specifies the title of the code file.</td>
</tr>
<tr>
<td>CODE_URL</td>
<td>Specifies the URL of the code file.</td>
</tr>
<tr>
<td>CONTENTS_NAME</td>
<td>Specifies the name of the contents file.</td>
</tr>
<tr>
<td>CONTENTS_TITLE</td>
<td>Specifies the title of the contents file.</td>
</tr>
<tr>
<td>CONTENTS_URL</td>
<td>Specifies the URL of the contents file.</td>
</tr>
<tr>
<td>DATA_NAME</td>
<td>Specifies the name of the data file.</td>
</tr>
<tr>
<td>DATA_TITLE</td>
<td>Specifies the title of the data file.</td>
</tr>
<tr>
<td>DATA_URL</td>
<td>Specifies the URL of the data file.</td>
</tr>
<tr>
<td>Event Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENCODING</td>
<td>Specifies the encoding of the output for converting text data into a numbering system that computers recognize.</td>
</tr>
<tr>
<td>FRAME_NAME</td>
<td>Specifies the name of the frame file.</td>
</tr>
<tr>
<td>FRAME_TITLE</td>
<td>Specifies the title of the frame file.</td>
</tr>
<tr>
<td>FRAME_URL</td>
<td>Specifies the URL of the frame file.</td>
</tr>
<tr>
<td>GRAPH_PATH_NAME</td>
<td>Specifies the path of the graph as specified by the ODS PATH statement.</td>
</tr>
<tr>
<td>GRAPH_PATH_URL</td>
<td>Specifies the URL of the graph.</td>
</tr>
<tr>
<td>NO_BOTTOM</td>
<td>is nonzero if you specified the NO_BOTTOM_MATTER option in the ODS MARKUP statement.</td>
</tr>
<tr>
<td>NO_TOP</td>
<td>is nonzero if you specified the NO_TOP_MATTER option in the ODS MARKUP statement.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>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.</td>
</tr>
<tr>
<td>PAGES_NAME</td>
<td>Specifies the name of the pages file.</td>
</tr>
<tr>
<td>PAGES_TITLE</td>
<td>Specifies the title of the pages file.</td>
</tr>
<tr>
<td>PAGES_URL</td>
<td>Specifies the URL of the pages file.</td>
</tr>
<tr>
<td>PATH</td>
<td>Specifies the path as set by an ODS statement.</td>
</tr>
<tr>
<td>PATH_NAME</td>
<td>Specifies the pathname.</td>
</tr>
<tr>
<td>PATH_URL</td>
<td>Specifies the path location.</td>
</tr>
<tr>
<td>STYLESHEET_NAME</td>
<td>Specifies the name of the style sheet file.</td>
</tr>
<tr>
<td>STYLESHEET_TITLE</td>
<td>Specifies the title of the style sheet file.</td>
</tr>
<tr>
<td>STYLESHEET_URL</td>
<td>Specifies the URL of the style sheet file.</td>
</tr>
<tr>
<td>TAGSET</td>
<td>Specifies the name of the current tagset.</td>
</tr>
<tr>
<td>Event Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TAGSET_ALIAS</td>
<td>Specifies the alias of the current tagset as given in the ODS MARKUP statement.</td>
</tr>
<tr>
<td>TITLE</td>
<td>Specifies the title from the ODS statement.</td>
</tr>
<tr>
<td>TRANTAB</td>
<td>Specifies the translation table name for character conversions.</td>
</tr>
</tbody>
</table>

Table 17.8  Table Variables

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLABEL</td>
<td>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 &quot;CONTENTS_LABEL= &quot;string&quot;</td>
</tr>
<tr>
<td>COLCOUNT</td>
<td>Specifies the number of columns in the current table.</td>
</tr>
<tr>
<td>COLEND_EA</td>
<td>Specifies the ending column number.</td>
</tr>
<tr>
<td>COLSPAN</td>
<td>Specifies the number of columns that the cell spans.</td>
</tr>
<tr>
<td>COLSTART</td>
<td>Specifies the column number where the cell starts.</td>
</tr>
<tr>
<td>DATA_ROW</td>
<td>Specifies that the current row is a data row.</td>
</tr>
<tr>
<td>IS_STACKED</td>
<td>Specifies that the columns are stacked.</td>
</tr>
<tr>
<td>ROW</td>
<td>Specifies the current table row, which includes headers.</td>
</tr>
<tr>
<td>ROWSPAN</td>
<td>Specifies the number of rows that the current cell spans.</td>
</tr>
<tr>
<td>SECTION</td>
<td>Specifies the header, body, or footer of the table.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Specifies the width. WIDTH is most commonly used for COLSPECS.</td>
</tr>
</tbody>
</table>
### Table 17.9 URL Variables

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOBASE</td>
<td>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.</td>
</tr>
<tr>
<td>TARGET</td>
<td>Specifies the target that is associated with the URL.</td>
</tr>
<tr>
<td>URL</td>
<td>Specifies a fully formed URL.</td>
</tr>
</tbody>
</table>

### Event Statement Conditions

Event statement conditions specify one or more conditions that must be true for a DEFINE EVENT statement to execute. An event statement condition must be preceded by a slash (/).

Event statement conditions have the following form:

```
define-event-statement / event-statement-condition(s);
```

- **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:  
```plaintext
put 'The foreground is blue!' nl/if cmp('blue', foreground);
```

CONTAINS (argument-1, argument-2)

searches the first argument for the second argument.

Example:  
```plaintext
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:  
```plaintext
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.

Restriction  
When you specify an IF condition with a single, user-defined variable, then the variable is evaluated to determine whether it has a value, according to the variable's type. A string variable type uses the length to determine existence, a numeric variable type uses value, and a dictionary array variable type uses the key (if there is a key specified, then the test is true).

Example:  
```plaintext
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 exists(foreground);
```

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:  
```plaintext
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;

Examples: TEMPLATE Procedure: Creating Markup Language Tagsets

Example 1: Creating a Tagset through Inheritance

Features:

- DEFINE TAGSET statement
- DEFINE EVENT statement
- PUT statement
- PARENT= attribute
- Other ODS features
  - ODS PATH statement
  - ODS MARKUP statement

Details

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

ods path sasuser.templat (update)
sashelp.tmplmst (read);

proc template;
   define tagset tagsets.mytags /store=sasuser.templat;
      parent=tagsets.chtml;
   done;
define event colspecs;
  put 'These are my new colspecs' nl;
end;

define event table;
  put '<p>' nl '<table>' nl;
finish:
  put '</table>'; nl;
end;

define event system_title;
end;
end;
run;
ods tagsets.mytags body='custom-tagset-filename.html';

proc print data=sashelp.class;
run;
ods tagsets.mytags close;

Program Description

**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.

```plaintext
ods path sasuser.templat (update)
sashelp.tmplmst (read);
```

```plaintext
proc template;
  define tagset tagsets.mytags /store=sasuser.templat;
    parent=tagsets.chtml;
end;
```

**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.

```plaintext
define event colspecs;
  put 'These are my new colspecs' nl;
end;

define event table;
  put '<p>' nl '<table>'; nl;
finish:
  put '</table>' nl;
end;

define event system_title;
end;
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;

To see the customized CHTML tags, view the source from the web browser: From the browser's toolbar, 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.

```sas
ods markup tagset=tagsets.chtml body='default-tagset-filename.html';

proc print data=sashelp.class;
run;

ods markup close;
```

To see the default CHTML tags, view the source from the web browser: From the browser's toolbar, select View ➔ Source.
### The SAS System

<table>
<thead>
<tr>
<th>Obs</th>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69.0</td>
<td>112.5</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>56.5</td>
<td>84.0</td>
</tr>
<tr>
<td>3</td>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>65.3</td>
<td>98.0</td>
</tr>
<tr>
<td>4</td>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
</tr>
<tr>
<td>5</td>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>6</td>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>83.0</td>
</tr>
<tr>
<td>7</td>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
</tr>
<tr>
<td>8</td>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>9</td>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84.0</td>
</tr>
<tr>
<td>10</td>
<td>John</td>
<td>M</td>
<td>12</td>
<td>59.0</td>
<td>99.5</td>
</tr>
<tr>
<td>11</td>
<td>Joyce</td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
</tr>
<tr>
<td>12</td>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>90.0</td>
</tr>
<tr>
<td>13</td>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>56.3</td>
<td>77.0</td>
</tr>
<tr>
<td>14</td>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>66.5</td>
<td>112.0</td>
</tr>
<tr>
<td>15</td>
<td>Philip</td>
<td>M</td>
<td>16</td>
<td>72.0</td>
<td>150.0</td>
</tr>
<tr>
<td>16</td>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>64.8</td>
<td>128.0</td>
</tr>
<tr>
<td>17</td>
<td>Ronald</td>
<td>M</td>
<td>15</td>
<td>67.0</td>
<td>133.0</td>
</tr>
<tr>
<td>18</td>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85.0</td>
</tr>
<tr>
<td>19</td>
<td>William</td>
<td>M</td>
<td>15</td>
<td>66.5</td>
<td>112.0</td>
</tr>
</tbody>
</table>
Example 2: Creating a Tagset By Copying a Tagset's Source

Features:
- SOURCE statement
- DEFINE TAGSET statement
- DEFINE EVENT statement

Details
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;
```
Partial Log: Default CSV Tagset That SAS Supplies

1 proc template;
   NOTE: Writing HTML Body file: sashtml.htm
2   source tagsets.csv;
   define tagset Tagsets.Csv;
   notes "This is the CSV definition";
   mvar  _CURRENCY_SYMBOL _DECIMAL_SEPARATOR _THOUSANDS_SEPARATOR _CURRENCY_AS_NUMBER
         _PERCENTAGE_AS_NUMBER _DELIMITER;
3   define event initialize;
4       trigger set_options;
5       trigger documentation;
6       trigger compile_regexp;
7       end;
8   define event options_set;
9       trigger set_options;
10      trigger documentation;
11     trigger compile_regexp;
12     end;
13   define event documentation;
14     trigger help /if cmp( $options["DOC"], "help");
15     trigger quick_reference /if cmp( $options["DOC"], "quick");
16     end;
17   define event help;
18     putlog
19     "==============================================================================";
20     putlog "The CSV Tagset Help Text.";
21     putlog "This Tagset/Destination creates output in comma separated value format.";
22     putlog "Numbers, Currency and percentages are correctly detected and show as numeric"
23     ...
...more lines of log...
24 define event warning;
25     break /if "$notes"
26     put VALUE NL;
27     end;
28     mapsub = */****/*;
29     map = "***;
30     registered_tm = "{r}";
31     trademark = "(tm)";
32     copyright = "(c)";
33     output_type = "csv";
34     stacked_columns = OFF;
35     end;
37 run;
38 NOTE: PROCEDURE TEMPLATE used (Total process time):
39    real time           13.14 seconds
40    cpu time            0.63 seconds
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.

```plaintext
proc template;
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;
```

Example 2: Creating a Tagset By Copying a Tagset's Source
To view the customized CSV tagset Tagsets.Mycsv, submit the following code:

```sas
proc template;
  source tagsets.mycsv;
run;
```

Log Output: Customized CSV Tagset Tagsets.Mycsv

You can view the tagset by going to Results ⇒ Templates ⇒ Sasuser.Templat ⇒ Mycsv.
Example 3: Creating a New Tagset

Features:

- PROC TEMPLATE features
- DEFINE TAGSET statement
- NOTES statement
- DEFINE EVENT statement
Details

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

```
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 ''; ndent;
   put '';
   put VALUE;
   put ''; nl;
end;

define event frequency;
   put '';
   put VALUE;
   put ''; nl;
   indent;
   put ''; nl;
end;

output_type = 'xml';
default_event = 'basic';
indent = 2;
split = '';
nobreakspace = '';
mapsub = '/</>/&/';
map = '<>&';
stacked_columns=off;
end;
run;

---

**Program Description**

**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';
   end;
run;
```

**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;
      nl;
      indent;
   finish:
      xdent;
      put nl;
      put '';
```
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;
  indent;
  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;

New Tagsets.NewlocTemplate Source

You can view the tagset by going to Results ➔ Templates ➔ Sasuser.Templat ➔ Newloc
Example 4: Executing Events Using the TRIGGER= Statement

Features:

DEFINE TAGSET statement
DEFINE EVENT statement
PUT statement
TRIGGER statement
Other ODS features
  ODS directory.tagset-name statement

Details
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.

```plaintext
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;
```

Output
To view the output Tagsets.Mytagset, open the file in a text editor.
Example 5: Indenting Output

Features:

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

Details

This example illustrates how to indent the output using a tagset. When you view a file with an extension of .xml in an XML-compliant browser, the browser ignores any indentation in the file in favor of its own indentation algorithm.

Program

Set the beginning indentation level and then proceed to increment the indentation 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 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
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;
run;
ods tagsets.mytagset2 file='custom-tagset-filename2.txt';
ods tagsets.mytagset2 close;
Example 6: Using Different Styles for Events

Features:
- DEFINE EVENT statement
- PUT statement
- TRIGGER statement
- STYLE= event attribute

Details
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.

```sas
define event Gnote;
start:
    put '<div>);
```
trigger align;
put '>'; put '<table'> 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;
Example 7: Modifying an Event to Include Other Style Sheets

Features:
- PROC TEMPLATE features
  - DEFINE EVENT statement
  - PUTQ statement

Details

The following program provides some example code that you can use to link a previously created style sheet to an event that you define.

Program

**Define an event that links to a style sheet.** This code defines an event that creates a link to a previously created style sheet instead of the style sheet that SAS generated.

```plaintext
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

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
Details

This example shows the difference between stacking data one column on top of another and placing data side by side. (For more information about stacked columns, see the "DEFINE TABLE Statement" on page 534.)

Program

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;

proc template;
   define tagset tagsets.myhtml;
      parent=tagsets.phtml;
      stacked_columns=no;
   end;
run;
ods tagsets.myhtml file='not_stacked.html';
proc standard print data=sashelp.class;
run;
ods _all_ close;

Program Description

Create a table template. The DEFINE TABLE statement creates the table template.

proc template;
   define table Base.Standard;
      notes 'Table template for PROC Standard.';
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.

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.

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.
Program

Create the same file but with stacked values. The STACKED_COLUMNS=YES statement shows the same values stacked in the SAS tagset PHTML.

```sas
ods phtml file='stacked.html';
proc standard print data=sashelp.class;
run;
ods _all_ close;
```
Example 8: Using the STACKED_COLUMNS Attribute in a Tagset

Output

Output 17.8  Output with Values Stacked One on Top of Another

The SAS System

The STANDARD Procedure

<table>
<thead>
<tr>
<th>Name</th>
<th>Mean/Std Dev</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>13.315789 1.492672</td>
<td>19</td>
</tr>
<tr>
<td>Height</td>
<td>62.336842 5.127075</td>
<td>19</td>
</tr>
<tr>
<td>Weight</td>
<td>100.026316 22.773933</td>
<td>19</td>
</tr>
</tbody>
</table>
PART 8

ODS Styles Reference

Chapter 18
*Overview* .................................................. 719

Chapter 19
*Style Templates* ............................................. 727

Chapter 20
*Style Elements* ............................................. 797

Chapter 21
*Style Attributes* .......................................... 827
Overview

Understanding Styles, Style Elements, and Style Attributes

The appearance of SAS output is controlled by ODS style templates (ODS styles). ODS styles are produced from compiled STYLE templates written in PROC TEMPLATE style syntax. An ODS style template is a collection of style elements that provides specific visual attributes for your SAS output.

- A style element is a named collection of style attributes that apply to a particular part of the output. Each area of ODS output has a style element name that is associated with it. The style element name specifies where the style attributes are applied. 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.

- A style attribute is a visual property, such as color, font properties, and line characteristics, that is defined in ODS with a reserved name and value. Style attributes are collectively referenced by a style element within a style template. Each style attribute specifies a value for one aspect of the presentation. For example, the BACKGROUND COLOR= 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 font or an italic font.

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

Available styles are in the SASHELP.TMPLMST item store. In SAS Enterprise Guide, the list of style sheets is shown by the Style Wizard. In batch mode or SAS Studio, you can display the list of available style templates by using the LIST statement in PROC TEMPLATE:

```
proc template;
  list styles / store=sashelp.tmplmst;
run;
```
For complete information about viewing ODS styles, see “Viewing ODS Styles Supplied by SAS” on page 727.

By default, HTML 4 output uses the HTMLBlue style template and HTML 5 output uses the HTMLEncore style template. To help you become familiar with styles, style elements, and style attributes, look at the relationship between them.

You can use the SOURCE statement in PROC TEMPLATE to display the structure of a style template. The following code prints the structure of the HTMLBlue style template to the SAS log:

```sas
proc template;
  source styles.HTMLBlue;
run;
```

The following figure illustrates the structure of a style. The figure shows the relationship between the style, the style elements, and the style attributes.
The following list corresponds to the numbered items in the preceding figure:

1 Styles.HTMLBlue is the style. Styles describe how to display presentation aspects (color, font, font size, and so on) of the SAS output. A style determines the overall appearance of the ODS documents that use it. The default style for HTML output is HTMLBlue. Each style consists of style elements.
You can create new styles with the "DEFINE STYLE Statement" on page 448. New styles can be created independently or from an existing style. You can use "PARENT= Statement" on page 454 to create a new style from an existing style. For complete documentation about ODS styles, see Chapter 19, "Style Templates," on page 727.

2 Header and Footer are examples of 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 styles and consist of one or more style attributes. Style elements can be user-defined or supplied by SAS. User-defined style elements can be created by the "STYLE Statement" on page 455.

Note: For a list of the default style elements used for HTML and markup languages and their inheritance, see Chapter 20, "Style Elements," on page 797.

3 BORDERCOLOR=, BACKGROUNDCOLOR=, and COLOR= are examples of style attributes. Style attributes specify a value for one aspect of the area of the output that its style element applies to. For example, the COLOR= attribute specifies the value cx112277 for the font color. For a list of style attributes supplied by SAS, see Chapter 21, "Style Attributes," on page 827.

Style attributes can be referenced with style references. See "style-reference" on page 894 for more information about style references.

The following table shows commonly used style attributes that you can set with the STYLE= option in PROC PRINT, PROC TABULATE, and PROC REPORT. Most of these attributes apply to parts of the table other than cells (for example, table borders and the lines between columns and rows). Note that not all attributes are valid in all destinations. For more information about these style attributes, their valid values, and their applicable destinations, see "Style Attributes Tables" on page 828.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>PROC REPORT Statement Area</th>
<th>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</th>
<th>PROC TABULATE STATEMENT TABLE</th>
<th>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</th>
<th>PROC PRINT TABLE location</th>
<th>PROC PRINT: all locations other than TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BACKGROUNDCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BACKGROUNDIMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Attribute</td>
<td>PROC REPORT STATEMENT REPORT Area</td>
<td>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</td>
<td>PROC TABULATE STATEMENT TABLE</td>
<td>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</td>
<td>PROC PRINT TABLE location</td>
<td>PROC PRINT: all locations other than TABLE</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>--------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>BORDERBOTTOMCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLORDATAK=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLORLIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPCOLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Attribute</td>
<td>PROC REPORT STATEMENT REPORT Area</td>
<td>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</td>
<td>PROC TABULATE STATEMENT VAR, CLASS, BOX, CLASSLEV, KEYWORD</td>
<td>PROC PRINT TABLE location</td>
<td>PROC PRINT: all locations other than TABLE</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------</td>
<td>----------------------------------------------------------</td>
<td>---------------------------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CELLPADDING=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CELLPACING=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CELLLWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CLASS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>COLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FLYOVER=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>FONT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>FONTFAMILY=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>FONTSIZE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>FONTSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>FONTWEIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>FONTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FRAME=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>HEIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>HREFTARGET=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>HTMLSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>NOBREAKSPACE=²</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OUTPUTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>POSTHTML=¹</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>POSTIMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>POSTTEXT=¹</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>PREHTML=¹</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>PREIMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X X X X</td>
<td></td>
</tr>
</tbody>
</table>
## Using Styles with 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 14, "TEMPLATE Procedure," on page 425.

The Base SAS reporting procedures, PROC PRINT, PROC REPORT, and PROC TABULATE, enable you to quickly analyze your data and organize it into easy-to-read tables. You can use the STYLE= option with these procedure statements to modify the appearance of your report. The STYLE= option enables you to make changes in sections of output without changing the default style for all of the output. You can customize specific sections of procedure output by specifying the STYLE= option in specific statements within the procedure.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>PROC REPORT STATION REPORT Area</th>
<th>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</th>
<th>PROC TABULATE STATEMENT TABLE</th>
<th>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</th>
<th>PROC PRINT TABLE location</th>
<th>PROC PRINT: all locations other than TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRETEXT=(^1)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PROTECTSPECIALCHARS=</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RULES=</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>TAGATTR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TEXTALIGN=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>URL=</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>VERTICALALIGN=</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>WIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

\(^1\) When you use these attributes in this location, they affect only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the foreground color or the font for the text that appears in the table, you must set the corresponding attribute in a location that affects the cells rather than the table. For complete documentation about style attributes and their values, see Chapter 21, "Style Attributes," on page 827.

\(^2\) To help prevent unexpected wrapping of long text strings when using PROC REPORT with the ODS RTF destination, set NOBREAKSPACE=OFF in a location that affects the LINE statement. The NOBREAKSPACE=OFF attribute must be set in the PROC REPORT code either on the LINE statement or on the PROC REPORT statement where style(lines) is specified.
Viewing ODS Styles Supplied by SAS

Over fifty ODS styles are available for use with ODS. To view the names of all of the style templates that are shipped with SAS, submit the following program. The style templates are all located in the Styles folder by default.

```sas
proc template;
   path sashelp.tmplmst;
   list styles;
run;
```
To view the source code of a specified style, submit the following code:

```sas
proc template;
  source styles.style-name;
run;
```

**Example Code 19.1**  Partial Source Code for the HTMLBlue Style

```sas
proc template;
  source styles.htmlBlue;
define style Styles.htmlBlue;
parent = styles.statistical;
class GraphColors /
  'gndata12' = cxECE8C4
  'gndata11' = cxDBD8F8
  'gndata10' = cxC6E4BF
  'gndata9' = cxE6CEAD
  'gndata8' = cxE5C1D4
  'gndata7' = cxDBC7E7
  'gndata6' = cxDDEEB5
  'gndata5' = cxDBC7E7
  'gndata4' = cxD5C6B4
  'gndata3' = cxB7D4D3
  'gndata2' = cxE7B3B4
  'gndata1' = cxBBC2DC
  'gndata' = cxC8C9CB
```
**Note:** If you are using SAS Studio, you do not need to specify the STYLE= option. You can go to Preferences ⇾ Results and change the style from the drop-down list for your selected destination.

---

**Program for Viewing Multiple Styles**

This program creates a sample report in HTML, PDF, and RTF of every style supplied by SAS. The output appears in your working directory. Although you can apply most SAS styles to any destination, SAS supplies one or more styles that are optimized to work with the output the destination creates. For a table of suggested ODS styles for each destination, see “Recommended Styles for ODS Destinations” in SAS Output Delivery System: Advanced Topics.

The table Gallery is created and is used in a subsequent DATA step to generate a list of each style for each destination. The destination link is a hyperlink.

```sas
ods _all_ close;

proc template;
  define table gallery;
    column libname memname style links;
    define libname ;
      blank_dups=on;
    end;
    define links;
      header = 'Samples';
      compute as '<a href="' || trim(style) || '.html">HTML</a> ' ||
        '<a href="' || trim(style) || '.pdf">PDF</a> ' ||
        '<a href="' || trim(style) || '.rtf">RTF</a>";
    end;
  end;
run;
```

The DATA step creates an index of all available styles supplied by SAS from the template store Sashelp.Tmplmst.

```sas
ods html file="index.html";
title "Index of all styles";
data _null_;  
  set sashelp.vstyle(where=(libname="SASHELP"));
  file print ods=(template='Gallery');
  put _ods_;
run;
ods html close;
```

The ODS destination statements create the output. You can add additional destinations by specifying the following statement for each destination:

```sas
ods destination file="&style..destination-extension" style=&style;
```

%macro generateods();
options nodate;
ods html file="&style..html" style=&style;
%endmacro;

```sas
%generateods();
```
The ODS NOPTITLE statement removes the procedure title.

```ods noptitle;
```

The ODS SELECT statement selects the Variables table for the gallery.

```ods select variables;
    proc contents data=sashelp.class;
    run;
    ods _all_ close;
%mend;
```

The ODS NORESULTS statement prevents an entry in the results window for each of the subsequent PROC CONTENTS steps that are generated.

```ods noresults;
```

This DATA step creates a sample of each style.

```data _null_;
    set sashelp.vstyle(where=(libname="SASHHELP"));
    call symputx('style', style);
    call execute('%generateods');
    run;
%mend;
```

The ODS RESULTS and ODS PREFERENCES statements set the ODS options back to defaults.

```ods results;
ods preferences;
```

---

**Table of Suggested ODS Styles**

With ODS, you can use any style with any output destination. However, for each destination, SAS supplies one or more styles that are optimized to work with the output the destination creates.

*Table 19.1 Recommended Styles for ODS Destinations*

<table>
<thead>
<tr>
<th>Destination</th>
<th>Recommended Styles</th>
<th>Default Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPUB</td>
<td>Daisy¹</td>
<td>Daisy</td>
</tr>
<tr>
<td></td>
<td>Moonflower</td>
<td></td>
</tr>
<tr>
<td>Printer family of statements</td>
<td>FancyPrinter</td>
<td>Pearl</td>
</tr>
<tr>
<td></td>
<td>FestivalPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrayscalePrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MeadowPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MonoChromePrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monospace</td>
<td></td>
</tr>
<tr>
<td>Destination</td>
<td>Recommended Styles</td>
<td>Default Style</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td></td>
<td>NormalPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pearl Printer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sapphire Printer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SasDocPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SeasidePrinter</td>
<td></td>
</tr>
<tr>
<td>RTF</td>
<td>RTF</td>
<td>RTF</td>
</tr>
<tr>
<td>TAGSETS.RTF</td>
<td>RTF</td>
<td>RTF</td>
</tr>
<tr>
<td>ODS destination for PowerPoint</td>
<td>PowerPointDark</td>
<td>PowerPointLight</td>
</tr>
<tr>
<td></td>
<td>PowerPointLight</td>
<td></td>
</tr>
<tr>
<td>SASREPORT for Enterprise Guide</td>
<td>EGDefault</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td></td>
<td>HTMLBlue</td>
<td></td>
</tr>
<tr>
<td>SASREPORT for Web Report Studio</td>
<td>Normal</td>
<td>Plateau</td>
</tr>
<tr>
<td></td>
<td>Plateau</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seaside</td>
<td></td>
</tr>
<tr>
<td>LISTING</td>
<td>Listing</td>
<td>Listing</td>
</tr>
<tr>
<td>HTML</td>
<td>Minimal</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td></td>
<td>EGDefault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seaside</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HTMLBlue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BarretttsBlue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BlockPrint</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dove</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dtree</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Festival</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gantt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Harvest</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HighContrast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal1a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal2</td>
<td></td>
</tr>
<tr>
<td>Destination</td>
<td>Recommended Styles</td>
<td>Default Style</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>Journal2a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Journal3a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Meadow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Netdraw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NoFontDefault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ocean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Plateau</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Raven</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SasWeb</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seaside</td>
<td></td>
</tr>
<tr>
<td></td>
<td>StatDoc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Statistical</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAGSETS.EXCELSXP</th>
<th>Default</th>
<th>Default</th>
</tr>
</thead>
</table>

1 The Moonflower style for ODS EPUB is designed for nighttime or low-light reading.
ODS Styles Gallery

EPUB Daisy Style

Output 19.1  EPUB Book Title Page
Chapter 19 / Style Templates

Output 19.2 EPUB Book, Page 1

The GLM Procedure

Class Level Information
Class Levels Values
Drug 3 A D F

Number of Observations Read 30
Number of Observations Used 30

Output 19.3 EPUB Book, Page 2

The GLM Procedure

Dependent Variable: PostTreatment

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type I SS</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>6</td>
<td>691,142,048</td>
<td>178,224,405</td>
<td>10.78</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>24</td>
<td>397,557,952</td>
<td>16,964,915</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected</td>
<td>29</td>
<td>1286,700,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
<td>1,475,400,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square 0.691509 Mean -0.511901 Root MSE 4.0790032 PostTreatment Mean 7.900000

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug</td>
<td>2</td>
<td>4.212,293</td>
<td>2.106,146</td>
<td>0.95</td>
<td>0.49</td>
<td>0.7355</td>
</tr>
<tr>
<td>PreTreatment</td>
<td>1</td>
<td>564,367,283</td>
<td>564,367,283</td>
<td>34.08</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>PreTreatmentDrug</td>
<td>2</td>
<td>19.64,905</td>
<td>9.822,222</td>
<td>0.55</td>
<td>0.9036</td>
<td></td>
</tr>
</tbody>
</table>
HTML Styles

You can view and modify the default HTML style by selecting **Tools** ↪ **Options** ↪ **Preferences** from the menu at the top of the main SAS window. Then open the **Results** tab. You can change the style by selecting a style from the **Style** drop-down menu. The settings in your Preferences window persist until you explicitly change them. The following display shows the **Results** tab with the new HTML style specified:

---

**Note:** If you are using SAS Studio, you do not need to specify the **STYLE=** option. You can go to **Preferences** ↪ **Results** and change the style from the drop-down list for your selected destination.
Figure 19.2 Changing the HTML Style with the Preferences Window
**Figure 19.3** Changing the HTML Style with SAS Studio

**Output 19.5** Analysis Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.6  Analysis Style: Graphs

---

**Graph 1**

Bar chart showing sales data for different regions:
- **BED**
- **CHAIR**
- **DESK**
- **SOFA**
- **TABLE**

Sales are categorized by region (EAST, WEST) and product type.

**Graph 2**

Scatter plot showing the relationship between height and weight with confidence and prediction limits.

- **Height**
- **Weight**

The plot includes 95% confidence limits and 95% prediction limits, along with a regression line.

---
Output 19.7  BarrettsBlue Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
### Output 19.8  BarretttsBlue Style: Graphs

<table>
<thead>
<tr>
<th>Bed</th>
<th>Chair</th>
<th>Desk</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Barrel Sales" /></td>
<td><img src="#" alt="Chair Sales" /></td>
<td><img src="#" alt="Desk Sales" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sofa</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Sofa Sales" /></td>
<td><img src="#" alt="Table Sales" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>East</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Eastern Sales" /></td>
<td><img src="#" alt="Western Sales" /></td>
<td><img src="#" alt="Western Sales" /></td>
</tr>
</tbody>
</table>

- **Predicted Sales**
- **Actual Sales**

---

### Weight vs. Height

- **95% Confidence Limits**
- **95% Prediction Limits**
- **Regression**

<table>
<thead>
<tr>
<th>Weight</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Weight vs. Height Graph" /></td>
<td><img src="#" alt="Height vs. Weight Graph" /></td>
</tr>
</tbody>
</table>
### Output 19.9  BlockPrint Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.10  BlockPrint Style: Graphs

The following style is recommended for generating accessible output when used with the ODS HTML5 destination.
Output 19.11  Daisy Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$346.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>

The following style is recommended for generating accessible output when used with the ODS HTML5 destination.
Output 19.12  Daisy Style: Graphs

![Bar Graphs](image)

![Scatter Plot](image)
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.14  Default Style: Graphs

- Bar charts showing sales for different regions (East, West) for Bed, Chair, Desk, Sofa, and Table.
- Line chart showing the relationship between height and weight with 95% confidence and prediction limits.
### Output 19.15  Dove Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$856.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.16  Dove Style: Graphs

[Bar charts showing sales for BED, CHAIR, DESK, SOFA, and TABLE in the EAST and WEST regions.]

[Scatter plot showing weight vs. height with 95% confidence limits and regression line.]
Output 19.17  Dir3e Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.18  Dtree Style: Graphs

The image shows a bar chart and a scatter plot. The bar chart displays sales data for different regions (East and West) and products (Bed, Chair, Sofa, Desk, Table) with predicted and actual sales lines. The scatter plot illustrates the relationship between height and weight, with confidence and prediction limits represented by shades and lines on the graph.
### Output 19.19  EGDefault Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.20  EGDefault Style: Graphs

[Bar charts showing sales for different regions and items: BED, CHAIR, DESK, SOFA, TABLE.]

[Graph showing relationship between weight and height.]

Legend:
- Predicted Sales
- Actual Sales
- 95% Confidence Limits
- 95% Prediction Limits
- Regression Line
Output 19.21  Festival Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$584.00</td>
</tr>
</tbody>
</table>
Output 19.22  Festival Style: Graphs

**Festival Style: Graphs**

- **Graph 1**: Comparison of predicted and actual sales across different regions (BED, CHAIR, DESK, SOFA, TABLE) showing varied sales figures with trends.
  - **Sales** vs **Region**
  - Legend: 
    - Predicted Sales
    - Actual Sales

- **Graph 2**: Scatter plot showing the relationship between weight and height with 95% confidence and prediction limits.
  - **Weight** vs **Height**
  - Legend: 
    - 95% Confidence Limits
    - 95% Prediction Limits
    - Regression Line
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
<tr>
<td>Obs</td>
<td>PRODUCT</td>
<td>REGION</td>
<td>ACTUAL</td>
<td>PREDICT</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$488.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$812.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.26  Harvest Style: Graphs

The following style is suitable for generating accessible output when used with the ODS HTML5 destination.
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>

The following style is suitable for generating accessible output when used with the ODS HTML5 destination.
HighContrast Style: Graphs

Output 19.28

HTMLBlue is the default style for the HTML output in the SAS Windowing Environment and SAS Studio.
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
In SAS 9.4M5, the HTMLEncore style uses the new Avenir Next for SAS fonts. For more information, see “TrueType Fonts Supplied by SAS” in SAS Language Reference: Concepts.
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$525.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$645.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.32  HTMLEncore: Graphs 1
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$825.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$812.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.34  Journal Style: Graphs

[Graphs showing sales for different regions (BED, CHAIR, DESK) with predicted and actual sales indicated.]

[Graph showing weight versus height with 95% confidence limits, 95% prediction limits, and regression line.]

[Bar charts for different items (BED, CHAIR, DESK, COFA, TABLE) with sales data for East and West regions.]
### Output 19.35  Meadow Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$808.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.36  Meadow Style: Graphs

[Graph showing bar charts for BED, CHAIR, DESK, SOFA, and TABLE, with sales on the y-axis and region (EAST, WEST) on the x-axis.]

[Graph showing a scatter plot with weight on the y-axis and height on the x-axis, including 95% confidence limits, 95% prediction limits, and regression line.]
## Output 19.37  Minimal Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.38  Minimal Style: Graphs
Output 19.39  Netdraw and NoFontDefault Styles: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.40  Netdraw and NoFontDefaul Styles: Graphs
### Output 19.41  Normal Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$808.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.42  Normal Style: Graphs
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.44  Ocean Style: Graphs
### Plateau Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$825.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$808.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$856.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$812.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.46  Plateau Style: Graphs

[Bar charts and line graphs showing sales for different regions and items, with predicted and actual sales plotted.]
### Output 19.47  Raven Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$584.00</td>
</tr>
</tbody>
</table>
Output 19.48  Raven Style: Graphs
Output 19.49  SasWeb Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$808.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.50  SasWeb Style: Graphs

The images depict bar charts showing sales data for different regions and items: BED, CHAIR, DESK for the East and West regions, and COFA, TABLE. The charts illustrate the predicted sales versus actual sales.

The second image is a scatter plot showing the relationship between height and weight with 95% confidence limits and prediction limits. The data points are connected by a regression line.
## Output 19.51  Seaside Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$808.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$812.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.52  Seaside Style: Graphs

The first graph shows the distribution of sales for different furniture items (BED, CHAIR, DESK, SOFA, TABLE) in the EAST and WEST regions. The y-axis represents sales in thousands of dollars, ranging from $0 to $20,000.

The second graph is a scatter plot showing the relationship between height and weight, with 95% confidence limits and 95% prediction limits. The regression line indicates the trend in the data.
<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$584.00</td>
</tr>
</tbody>
</table>
Output 19.54  StatDoc Style: Graphs

**Graph 1:**
- **Title:** Sales by Region
- **Y-axis:** Sales ($0k - $20k)
- **X-axis:** Region (East, West)
- **Legend:**
  - Predicted Sales
  - Actual Sales

**Graph 2:**
- **Title:** Weight vs. Height
- **Y-axis:** Weight (25 - 150)
- **X-axis:** Height (50 - 70)
- **Legend:**
  - 95% Confidence Limits
  - 95% Prediction Limits
  - Regression Line
Output 19.55  Statistical Style: Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 19.56  Statistical Style: Graphs
## Printer Styles

**Output 19.57  FancyPrinter Style**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,306</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

**Output 19.58  FestivalPrinter Style**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,306</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
### Output 19.59  GrayscalePrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

### Output 19.60  MeadowPrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
### Output 19.61 MonoChromePrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A41.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

### Output 19.62 Monospace Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A41.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
Pearl is the default style for PRINTER output.

### Output 19.63  NormalPrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,259</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$97,975</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,569</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,502</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,045</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$39,765</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,289</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$36,982</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

### Output 19.64  Pearl Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,259</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$97,975</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,569</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,502</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,045</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$39,765</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,289</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,952</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
## Output 19.65  Sapphire Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

## Output 19.66  SasDocPrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
Output 19.67  Seaside Printer Style

Styles for the ODS Destination for PowerPoint

PowerPointLight is the default style for output created by the ODS destination for PowerPoint.

Output 19.68  PowerPointLight Style
### Style for the ODS Destination for Excel

**Output 19.70** Excel Style

<table>
<thead>
<tr>
<th>Region</th>
<th>Division</th>
<th>Product type</th>
<th>Total Predicted Sales</th>
<th>Total Actual Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EAST</strong></td>
<td>CONSUMER</td>
<td>FURNITURE</td>
<td>$11,081</td>
<td>$12,483</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>$21,939</td>
<td>$16,991</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>FURNITURE</td>
<td>$12,972</td>
<td>$14,467</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>$16,434</td>
<td>$20,189</td>
</tr>
<tr>
<td></td>
<td>Division Total</td>
<td></td>
<td>$62,426</td>
<td>$64,130</td>
</tr>
<tr>
<td><strong>WEST</strong></td>
<td>CONSUMER</td>
<td>FURNITURE</td>
<td>$10,286</td>
<td>$10,380</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>$16,042</td>
<td>$16,371</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>FURNITURE</td>
<td>$12,816</td>
<td>$11,234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>$17,759</td>
<td>$18,905</td>
</tr>
<tr>
<td></td>
<td>Division Total</td>
<td></td>
<td>$56,903</td>
<td>$56,890</td>
</tr>
<tr>
<td></td>
<td><strong>Grand Total</strong></td>
<td></td>
<td>$119,329</td>
<td>$121,020</td>
</tr>
</tbody>
</table>
Style Elements

ODS Style Elements

Overview

The following tables lists all the style elements available for ODS style definitions. The tables provides a brief description of each style element and indicates the style elements from which it inherits its attributes.

Abstract style elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles do not appear in the output of destinations that generate a style sheet.
### Style Elements Affecting Documents

#### Table 20.1  Style Elements Affecting Documents

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>Controls the various document bodies. This generally includes things like the page background color and page margins. Note: This is an abstract style element.</td>
<td>Container</td>
</tr>
<tr>
<td>Body</td>
<td>Controls the Body file. The Body file is created by the BODY= option in any ODS Markup Family Destination.</td>
<td>Document</td>
</tr>
<tr>
<td>Frame</td>
<td>Controls the Frame file for HTML. The Frame file is created by the FRAME= option in any ODS Markup Family Destination.</td>
<td>Document</td>
</tr>
<tr>
<td>Contents</td>
<td>Controls the Contents file. The Contents file is created by the CONTENTS= option in any ODS Markup Family Destination.</td>
<td>Document</td>
</tr>
<tr>
<td>Pages</td>
<td>Controls the Page file. The Pages file is created by the PAGE= option in any ODS Markup Family Destination.</td>
<td>Document</td>
</tr>
</tbody>
</table>

### Style Elements Affecting Tables of Contents

#### Table 20.2  Style Elements Affecting Tables of Contents and Tables of Pages

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>Controls the Contents file</td>
<td>Document</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>IndexItem</td>
<td>Controls list items and folders for Contents and Pages</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>Note: This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>ContentFolder</td>
<td>Controls the folders in the Contents file</td>
<td>IndexItem</td>
</tr>
<tr>
<td>ByContentFolder</td>
<td>Controls the bygroup folders in the Contents file</td>
<td>ContentFolder</td>
</tr>
<tr>
<td>ContentItem</td>
<td>Controls the items in the Contents file</td>
<td>IndexItem</td>
</tr>
<tr>
<td>Index</td>
<td>Controls miscellaneous Contents and Pages components</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>Note: This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>IndexProcName</td>
<td>Controls the PROC name in the Contents and Pages files</td>
<td>Index</td>
</tr>
<tr>
<td></td>
<td>Note: This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>ContentProcName</td>
<td>Controls the PROC name in the Contents file</td>
<td>IndexProcName</td>
</tr>
<tr>
<td>ContentProcLabel</td>
<td>Controls the PROC label in the Contents file</td>
<td>ContentProcName</td>
</tr>
<tr>
<td>IndexAction</td>
<td>Determines what happens on mouse-over events for folders and items (HTML only)</td>
<td>IndexItem</td>
</tr>
<tr>
<td>FolderAction</td>
<td>Determines what happens on mouse-over events for folders (HTML only)</td>
<td>IndexAction</td>
</tr>
<tr>
<td>IndexTitle</td>
<td>Controls the title of Contents and Pages files</td>
<td>Index</td>
</tr>
<tr>
<td></td>
<td>Note: This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>ContentTitle</td>
<td>Controls the title of the Contents file</td>
<td>IndexTitle</td>
</tr>
</tbody>
</table>
Style Elements Affecting Pages

The PAGE= option in applicable MARKUP family destinations creates the page file. See PAGE= option.

Table 20.3  Style Elements Affecting Pages

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pages</td>
<td>Controls the Page file. The Pages file is created by the PAGE= option.</td>
<td>Document</td>
</tr>
<tr>
<td>Date</td>
<td>Controls how date fields look</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>PagesDate</td>
<td>Controls the date in the Pages file</td>
<td>Date</td>
</tr>
<tr>
<td>PagesTitle</td>
<td>Controls the title of the Pages file</td>
<td>IndexTitle</td>
</tr>
<tr>
<td>PagesItem</td>
<td>Controls the leafnode item in the Pages file</td>
<td>IndexItem</td>
</tr>
<tr>
<td>PagesProcName</td>
<td>Controls the proc name in the Pages file</td>
<td>IndexProcName</td>
</tr>
<tr>
<td>PagesProcLabel</td>
<td>Controls the proc label in the Pages file</td>
<td>PagesProcName</td>
</tr>
</tbody>
</table>

Style Elements Affecting Frames

The FRAME= option in ODS MARKUP destinations creates the frame file. See FRAME= option.

Table 20.4  Style Elements Affecting Frame

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame</td>
<td>Controls the Frame file for HTML</td>
<td>Document</td>
</tr>
</tbody>
</table>
Style Elements Affecting the Body File

The BODY style element inherits from the DOCUMENT style element. The BODY= option in applicable Markup Family destinations creates the body file. See BODY= option.

The following style elements affect the body of an HTML document:

- Tables on page 801
- Text on page 804
- Layouts on page 807

Style Elements Affecting Tables

**Table 20.5  Style Elements Affecting Data Cells in Tables**

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Default style for data cells</td>
<td>Cell</td>
</tr>
<tr>
<td>Cell</td>
<td>Controls data, header, and footer cells</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>DataFixed</td>
<td>Default style for data cells that request a fixed font</td>
<td>Data</td>
</tr>
<tr>
<td>DataEmpty</td>
<td>Controls empty data cells</td>
<td>Data</td>
</tr>
<tr>
<td>DataEmphasis</td>
<td>Controls emphasized data cells</td>
<td>Data</td>
</tr>
<tr>
<td>DataEmphasisFixed</td>
<td>Controls emphasized data cells that request a fixed font</td>
<td>DataEmphasis</td>
</tr>
<tr>
<td>DataStrong</td>
<td>Controls strong (more emphasized) data cells</td>
<td>Data</td>
</tr>
<tr>
<td>DataStrongFixed</td>
<td>Controls strong (more emphasized) data cells that request a fixed font</td>
<td>DataStrong</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Header</td>
<td>Controls the headers of a table</td>
<td>HeadersAndFooters</td>
</tr>
<tr>
<td>HeaderFixed</td>
<td>Controls the header of a table that request a fixed font</td>
<td>Header</td>
</tr>
<tr>
<td>HeaderEmphasis</td>
<td>Controls emphasized table header cells that request a fixed font</td>
<td>Header</td>
</tr>
<tr>
<td>HeaderEmphasisFixed</td>
<td>Controls emphasized table header cells that request a fixed font</td>
<td>HeaderEmphasis</td>
</tr>
<tr>
<td>HeaderStrong</td>
<td>Controls strong (more emphasized) table header cells</td>
<td>Header</td>
</tr>
<tr>
<td>HeaderStrongFixed</td>
<td>Controls strong (more emphasized) table header cells</td>
<td>HeaderStrong</td>
</tr>
<tr>
<td>HeaderEmpty</td>
<td>Controls empty table header cells</td>
<td>Header</td>
</tr>
<tr>
<td>RowHeader</td>
<td>Controls row headers</td>
<td>Header</td>
</tr>
<tr>
<td>RowHeaderFixed</td>
<td>Controls row headers that request a fixed font</td>
<td>RowHeader</td>
</tr>
<tr>
<td>RowHeaderEmpty</td>
<td>Controls empty row headers</td>
<td>RowHeader</td>
</tr>
<tr>
<td>RowHeaderEmphasis</td>
<td>Controls emphasized row headers</td>
<td>RowHeader</td>
</tr>
<tr>
<td>RowHeaderEmphasisFixed</td>
<td>Controls emphasized row headers that request a fixed font</td>
<td>RowHeaderEmphasis</td>
</tr>
<tr>
<td>RowHeaderStrong</td>
<td>Controls strong (more emphasized) row headers</td>
<td>RowHeader</td>
</tr>
<tr>
<td>RowHeaderStrongFixed</td>
<td>Controls strong (more emphasized) row headers that request a fixed font</td>
<td>RowHeaderStrong</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Footer</td>
<td>Controls table footers</td>
<td>HeadersAndFooters</td>
</tr>
<tr>
<td>FooterFixed</td>
<td>Controls table footers that request a fixed font</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterEmpty</td>
<td>Controls empty table footers</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterEmphasis</td>
<td>Controls emphasized table footers</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterEmphasisFixed</td>
<td>Controls emphasized table footers that request a fixed font</td>
<td>FooterEmphasis</td>
</tr>
<tr>
<td>FooterStrong</td>
<td>Controls strong (more emphasized) table footers</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterStrongFixed</td>
<td>Controls strong (more emphasized) table footers that request a fixed font</td>
<td>FooterStrong</td>
</tr>
<tr>
<td>RowFooter</td>
<td>Controls a row footer (label)</td>
<td>Footer</td>
</tr>
<tr>
<td>RowFooterFixed</td>
<td>Controls a row footer (label) that request a fixed font</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterEmpty</td>
<td>Controls an empty row footer (label)</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterEmphasis</td>
<td>Controls an emphasized row footer (label)</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterEmphasisFixed</td>
<td>Controls an emphasized row footer (label) that request a fixed font</td>
<td>RowFooterEmphasis</td>
</tr>
<tr>
<td>RowFooterStrong</td>
<td>Controls a strong (more emphasized) row footer (label)</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterStrongFixed</td>
<td>Controls a strong (more emphasized) row footer (label) that request a fixed font</td>
<td>RowFooterStrong</td>
</tr>
</tbody>
</table>
# Style Elements Affecting Text

Table 20.8  Elements affecting Text

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>BodyDate</td>
<td>Controls the date field in the Body file</td>
<td>ContentsDate</td>
</tr>
<tr>
<td>SASDate</td>
<td>Controls the date field in the Body file when SASDATE option is in effect.</td>
<td>BodyDate</td>
</tr>
<tr>
<td></td>
<td>This Element is valid only in the ODS destination for PowerPoint.</td>
<td></td>
</tr>
<tr>
<td>PageNo</td>
<td>Controls page numbers for paginated destinations (RTF, TAGSETS.RTF, Printer,</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td></td>
<td>PowerPoint)</td>
<td></td>
</tr>
<tr>
<td>SysTitleAndFooterContainer</td>
<td>Controls the container for system page title and system page footer.</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>This element is usually used to add borders around a title.</td>
<td></td>
</tr>
<tr>
<td>TitlesAndFooters</td>
<td>Controls system page title text and system page footer text</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>Note: This is an abstract style element.</td>
<td></td>
</tr>
<tr>
<td>SystemTitle</td>
<td>Controls system title text</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>SystemTitle2-10</td>
<td>Controls SystemTitle text for the corresponding SystemTitle value. For</td>
<td>SystemTitle-SystemTitle9</td>
</tr>
<tr>
<td></td>
<td>example, SystemTitle3 controls SystemTitle3 text.</td>
<td></td>
</tr>
<tr>
<td>SystemFooter</td>
<td>Controls system footer text</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>SystemFooter2-10</td>
<td>Controls SystemFooter text for the corresponding SystemFooter value. For</td>
<td>SystemFooter-SystemFooter9</td>
</tr>
<tr>
<td></td>
<td>example, SystemFooter3 controls SystemFooter3.</td>
<td></td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>controls SystemFooter3 text.</td>
<td></td>
</tr>
<tr>
<td>BylineContainer</td>
<td>Controls the container for the byline. This is generally used to add borders to a byline.</td>
<td>Container</td>
</tr>
<tr>
<td>Byline</td>
<td>Controls byline text</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>Parskip</td>
<td>Controls space between tables in RTF output</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>ProcTitle</td>
<td>Controls procedure title text</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>ProcTitleFixed</td>
<td>Controls procedure title text that requests a fixed font</td>
<td>ProcTitle</td>
</tr>
<tr>
<td>UserText</td>
<td>Controls the ODS TEXT= style</td>
<td>Note</td>
</tr>
<tr>
<td>PrePage</td>
<td>Controls the ODS RTF/MEASURED PREPAGE= style</td>
<td>Note</td>
</tr>
<tr>
<td>Note</td>
<td>Controls the container for note banners and note contents</td>
<td>Container</td>
</tr>
<tr>
<td>NoteBanner</td>
<td>Controls the banner for Note</td>
<td>Note</td>
</tr>
<tr>
<td>NoteContent</td>
<td>Controls the contents for Note</td>
<td>Note</td>
</tr>
<tr>
<td>NoteContentFixed</td>
<td>Controls the contents for Note. Fixed font.</td>
<td>NoteContent</td>
</tr>
<tr>
<td>WarnBanner</td>
<td>Controls the banner for Warnings</td>
<td>Note</td>
</tr>
<tr>
<td>WarnContent</td>
<td>Controls the contents of Warnings</td>
<td>Note</td>
</tr>
<tr>
<td>WarnContentFixed</td>
<td>Controls the contents for Warnings. Fixed font.</td>
<td>WarnContent</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>ErrorBanner</td>
<td>Controls the banner for Errors</td>
<td>Note</td>
</tr>
<tr>
<td>ErrorContent</td>
<td>Controls the contents of Errors</td>
<td>Note</td>
</tr>
<tr>
<td>ErrorContentFixed</td>
<td>Controls the contents for Errors. Fixed font.</td>
<td>ErrorContent</td>
</tr>
<tr>
<td>FatalBanner</td>
<td>Controls the banner for Fatal</td>
<td>Note</td>
</tr>
<tr>
<td>FatalContent</td>
<td>Controls the contents of Fatal</td>
<td>Note</td>
</tr>
<tr>
<td>FatalContentFixed</td>
<td>Controls the contents for Fatal. Fixed font.</td>
<td>FatalContent</td>
</tr>
<tr>
<td>Paragraph</td>
<td>Controls paragraphs in text</td>
<td>Container</td>
</tr>
<tr>
<td>Heading1-Heading10</td>
<td>Controls Heading text for the corresponding Headingn value. For example, Heading2 controls the Heading2 text.</td>
<td>NormalText</td>
</tr>
<tr>
<td>List</td>
<td>Controls both unordered and ordered lists</td>
<td>Container</td>
</tr>
<tr>
<td>List2-List10</td>
<td>Controls nested bulleted lists at the corresponding Listn level. For example, List3 controls the third level bulleted list.</td>
<td>List-List9</td>
</tr>
<tr>
<td>ListItem</td>
<td>Controls bulleted list items</td>
<td>Container</td>
</tr>
<tr>
<td>ListItem2-ListItem10</td>
<td>Controls the nested bulleted list items at the corresponding ListItemn level. For example, ListItem3 controls the third level bulleted list item.</td>
<td>ListItem-ListItem9</td>
</tr>
<tr>
<td>Pagebreak</td>
<td>Controls page breaks in the text</td>
<td>No inheritance</td>
</tr>
<tr>
<td>LineContent</td>
<td>Controls the text generated by the LINE</td>
<td>No inheritance</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>statement in PROC REPORT</td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td>Controls the color of unvisited link text</td>
<td>No inheritance</td>
</tr>
<tr>
<td>ActiveLink</td>
<td>Controls the color of active (clicked) link text</td>
<td>No inheritance</td>
</tr>
<tr>
<td>FocusLink</td>
<td>Controls the color of the link-focus indicator</td>
<td>No inheritance</td>
</tr>
<tr>
<td></td>
<td>Accessibility note: For an example that uses the FocusLink style element, see Enhancing the Appearance of the Link-Focus Indicator.</td>
<td></td>
</tr>
<tr>
<td>VisitedLink</td>
<td>Controls the color of visited link text</td>
<td>No inheritance</td>
</tr>
</tbody>
</table>

## Style Elements Affecting Layout

**Table 20.9 Elements Affecting Layout**

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayoutContainer</td>
<td>Container for Layout</td>
<td>No inheritance</td>
</tr>
<tr>
<td>LayoutRegion</td>
<td>Region style for Layout cells</td>
<td>LayoutContainer</td>
</tr>
</tbody>
</table>

**Note:** Layouts and regions are created by the ODS LAYOUT and ODS REGION statements. See ODS LAYOUT ABSOLUTE Statement, and ODS REGION Statement, Absolute

## Style Elements Affecting Graphs

For more information about Style Elements that affect graphs, see “Style Elements for Use with ODS Graphics” in SAS ODS Graphics: Procedures Guide.
## Miscellaneous Elements

### Table 20.10  Miscellaneous Style Elements

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Controls all container-oriented elements</td>
<td></td>
</tr>
<tr>
<td>Continued</td>
<td>Controls continued flag when a table breaks across a page (paginated destinations only)</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>ExtendedPage</td>
<td>Displays a message when the page does not fit (Printer only)</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>StartUpFunction</td>
<td>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.</td>
<td>No inheritance</td>
</tr>
<tr>
<td>ShutDownFunction</td>
<td>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.</td>
<td>No inheritance</td>
</tr>
<tr>
<td>UserText</td>
<td>Controls the ODS TEXT= style</td>
<td>Note</td>
</tr>
</tbody>
</table>

### 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 ODS Graphics output where a compiled ODS template of type STATGRAPH is used to produce graphical output. Supplied templates are stored in
Sashelp.Tmplmst. SAS/GRAH device drivers and some global statements such as SYMBOL, PATTERN, AXIS, and LEGEND have no affect. Common ODS Graphics procedures that produce template-based graphics are SGPLOT, SGPANEL, SGSCATTER, 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 that 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 “About Style Attributes” on page 459.

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphAnnoLine</td>
<td>Annotation lines and arrows, and outlines for closed annotation shapes such as circles and squares.</td>
<td>ContrastColor, LineStyle, LineThickness</td>
</tr>
<tr>
<td>GraphAnnoShape</td>
<td>Fill for closed annotation shapes such as circles and squares, and the default annotation marker in ODS Graphics Editor.</td>
<td>Color, MarkerSize, MarkerSymbol, Transparency</td>
</tr>
<tr>
<td>GraphAnnoText</td>
<td>Annotation text</td>
<td>Color, Font or font-attributes¹</td>
</tr>
<tr>
<td>GraphAxisLines</td>
<td>X-, Y-, and Z-axis lines</td>
<td>ContrastColor, LineStyle, LineThickness, TickDisplay</td>
</tr>
<tr>
<td>GraphBackground</td>
<td>Background of the graph</td>
<td>Color, Transparency</td>
</tr>
<tr>
<td>GraphBorderLines</td>
<td>Border line around the graph and around the graph legend</td>
<td>ContrastColor, LineStyle, LineThickness</td>
</tr>
<tr>
<td>GraphDataText</td>
<td>Text font and color for point and line labels</td>
<td>Color, Font or font-attributes¹</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphFootnoteText</td>
<td>Text font and color for footnote(s)</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font or font-attributes(^1)</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayOpts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphHeaderBackground</td>
<td>Background color of the cell headers in a data-driven lattice or data-driven panel graph</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphLabelText</td>
<td>Text font and color for axis labels and legend titles</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font or font-attributes(^1)</td>
</tr>
<tr>
<td>GraphLegendBackground</td>
<td>Background color of the legend</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphMinorGridLines</td>
<td>Appearance of the horizontal and vertical minor grid lines.</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayOpts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphOutlines</td>
<td>Outline properties for fill areas such as bars, pie slices, box plots, ellipses, and histograms</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphReference</td>
<td>Horizontal and vertical reference lines and drop lines</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphTitleText</td>
<td>Text font and color for title(s)</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font or font-attributes(^1)</td>
</tr>
<tr>
<td>GraphUnicodeText</td>
<td>Text font for Unicode values</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font or font-attributes(^1)</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>GraphValueText</td>
<td>Text font and color for axis tick values and legend values</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or font-attributes¹</td>
</tr>
<tr>
<td>GraphWalls</td>
<td>Vertical wall(s) bounded by axes</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FrameBorder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphBoxMean</td>
<td>Marker for mean</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>GraphBoxMedian</td>
<td>Line for median</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphBoxWhisker</td>
<td>Box whiskers and serifs</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphConfidence</td>
<td>Primary confidence lines and bands, colors for bands and lines</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphConfidence2</td>
<td>Secondary confidence lines and bands, color for bands, and</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td>contrast color for lines</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
</tbody>
</table>

¹ Font-attributes can be one or more of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.

Table 20.12  Style Elements Affecting Graphical Data Representation
<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphConnectLine</td>
<td>Line for connecting boxes or bars</td>
<td>Transparency, ContrastColor, LineStyle, LineThickness</td>
</tr>
<tr>
<td>GraphCutLine</td>
<td>Cutline attributes for a dendogram</td>
<td>Color, LineStyle</td>
</tr>
<tr>
<td>GraphDataDefault</td>
<td>Primitives related to non-grouped data items, colors for filled areas, markers, and lines</td>
<td>Color, ContrastColor, EndColor, LineStyle, LineThickness, NeutralColor, MarkerSymbol, MarkerSize, StartColor</td>
</tr>
<tr>
<td>GraphError</td>
<td>Error line or error bar fill, ContrastColor for lines, Color for bar fill</td>
<td>CapStyle, Color, ContrastColor, LineStyle</td>
</tr>
<tr>
<td>GraphFit</td>
<td>Primary fit lines such as a normal density curve</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphFit2</td>
<td>Secondary fit lines such as a kernel density curve</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphFinal</td>
<td>Final data for the waterfall chart. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>GraphInitial</td>
<td>Initial data for the waterfall chart. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphMissing</td>
<td>Properties for graph items representing missing values</td>
<td>Color, ContrastColor, FillPattern, LineStyle, LineThickness, MarkerSize, MarkerSymbol, Transparency</td>
</tr>
<tr>
<td>GraphOther</td>
<td>Other data for the graph. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphOverflow</td>
<td>Overflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphOutlier</td>
<td>Outlier data for the graph</td>
<td>ContrastColor, Color, MarkerSize</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>GraphPrediction</td>
<td>Prediction lines</td>
<td>ContrastColor, Color, LineStyle, LineThickness, MarkerSize, MarkerSymbol, Transparency</td>
</tr>
<tr>
<td>GraphPredictionLimits</td>
<td>Fills for prediction limits</td>
<td>ContrastColor, Color, LineStyle, LineThickness, MarkerSize, MarkerSymbol, Transparency</td>
</tr>
<tr>
<td>GraphSelection</td>
<td>For interactive graphs, visual properties of selected item. Color for selected fill area, ContrastColor for selected marker or line.</td>
<td>ContrastColor, Color, LineStyle, LineThickness, MarkerSize, MarkerSymbol, Transparency</td>
</tr>
<tr>
<td>GraphUnderflow</td>
<td>Underflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>ThreeColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>EndColor, NeutralColor, StartColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>ThreeColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NeutralColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>TwoColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>TwoColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
</tbody>
</table>

1 Attribute FillPattern is available in style element GraphMissing starting with SAS 9.4M5.

**Table 20.13  Graphical Style Elements: Data Related (Grouped)**

<table>
<thead>
<tr>
<th>Style Elements</th>
<th>Portion of Graph Affected</th>
<th>Attributes Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphData1</td>
<td>Primitives related to the first seven grouped data items. Color applies to filled areas.</td>
<td>Color</td>
</tr>
<tr>
<td>GraphData2</td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td>GraphData3</td>
<td></td>
<td>FillPattern¹</td>
</tr>
<tr>
<td>GraphData4</td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td>GraphData5</td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>GraphData6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData8</td>
<td>Primitives related to the 8th through 11th grouped data items.</td>
<td>Color</td>
</tr>
<tr>
<td>GraphData9</td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td>GraphData10</td>
<td></td>
<td>FillPattern¹</td>
</tr>
<tr>
<td>GraphData11</td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td>GraphData12</td>
<td>Primitives related to the 12th grouped data item.</td>
<td>Color</td>
</tr>
<tr>
<td>GraphData13</td>
<td>Primitives related to the 13th through 15th grouped data items.</td>
<td>FillPattern¹</td>
</tr>
<tr>
<td>GraphData14</td>
<td></td>
<td>MarkerSymbol</td>
</tr>
</tbody>
</table>
### Style Elements

<table>
<thead>
<tr>
<th>Style Elements</th>
<th>Portion of Graph Affected</th>
<th>Attributes Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphData15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Prior to **SAS 9.4M5**, style attribute FillPattern is available only with the JOURNAL2, JOURNAL3, and MONOCHROMEPRINTER styles. Starting with **SAS 9.4M5**, style attribute FillPattern in GraphData1–GraphData11 is also available with the DEFAULT ODS style and all styles that are derived from it.

2. Style attribute MarkerSymbol in these style elements is defined for styles JOURNAL2, JOURNAL3, and MONOCHROMEPRINTER only.

3. Style elements GraphData13–GraphData15 are available only with the JOURNAL2, JOURNAL3, and MONOCHROMEPRINTER styles.

### Table 20.14  Display Style Elements

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphAltBlock</td>
<td>Alternate fill color for block plots</td>
<td>Color</td>
<td>GraphColors(&quot;gablock&quot;)</td>
</tr>
<tr>
<td>GraphBand</td>
<td>Display options for confidence bands</td>
<td>DisplayOpts</td>
<td>&quot;Fill fillpattern outline&quot;</td>
</tr>
<tr>
<td>GraphBar</td>
<td>Display options for bar charts</td>
<td>DisplayOpts</td>
<td>&quot;Connect fill fillpattern outline&quot;</td>
</tr>
<tr>
<td>GraphBlock</td>
<td>Fill color for block plots</td>
<td>Color</td>
<td>GraphColors(&quot;gblock&quot;)</td>
</tr>
<tr>
<td>GraphBox</td>
<td>Display options for box plots</td>
<td>DisplayOpts</td>
<td>&quot;Caps connect fill fillpattern mean median notches outliers&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CapStyle</td>
<td>&quot;Serif&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connect</td>
<td>&quot;Mean&quot;</td>
</tr>
<tr>
<td>GraphContour</td>
<td>Display options for contours</td>
<td>DisplayOpts</td>
<td>&quot;LabeledLineGradient&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
<td>GraphColors('gramp3cend')</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NeutralColor</td>
<td>GraphColors('gramp3cnutral')</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
<td>GraphColors('gramp3cstart')</td>
</tr>
<tr>
<td>GraphEllipse</td>
<td>Display options for confidence ellipses</td>
<td>DisplayOpts</td>
<td>&quot;Fill fillpattern outline&quot;</td>
</tr>
<tr>
<td>GraphHighLow</td>
<td>Display options for high-low plots</td>
<td>DisplayOpts</td>
<td>&quot;Fill fillpattern outline&quot;</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
<td>Possible Values</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>GraphHistogram</td>
<td>Display options for histograms</td>
<td>DisplayOpts</td>
<td>&quot;Fill fillpattern outline&quot; 1</td>
</tr>
<tr>
<td>GraphPolygon 2</td>
<td>Display options for polygon plots</td>
<td>DisplayOpts</td>
<td>&quot;Fill fillpattern outline&quot;</td>
</tr>
<tr>
<td>GraphSkins</td>
<td>Display skins</td>
<td>DataSkin</td>
<td>&quot;Crisp&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Gloss&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Matte&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;None&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Pressed&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KpiSkin</td>
<td>&quot;Basic&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Modern&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;None&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Onyx&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Satin&quot;</td>
</tr>
</tbody>
</table>

1 Display option FillPattern is available starting with SAS 9.4M5.
2 Style elements GraphHighLow and GraphPolygon are valid starting with SAS 9.4M5.

### 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 device 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.

For complete documentation on the style attributes that can be specified in each style element, see "About Style Attributes" on page 459.

**Note:** These style elements affect device-based graphics only when the GSTYLE system option is in effect (this 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 System Options: Reference*. 
<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DropShadowStyle</td>
<td>Used with text types</td>
<td>Color</td>
</tr>
<tr>
<td>Graph</td>
<td>Graph size and outer border appearance</td>
<td>OutputWidth, OutputHeight, BorderColor, BorderWidth,_CellPadding, CellSpacing</td>
</tr>
<tr>
<td>GraphAxisLines</td>
<td>X, Y, and Z-axis lines</td>
<td>Color, LineStyle, LineThickness</td>
</tr>
<tr>
<td>GraphBackground</td>
<td>Background of the graph</td>
<td>Transparency, BackgroundColor, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image, VerticalAlign, TextAlign</td>
</tr>
<tr>
<td>GraphBorderLines</td>
<td>Border around graph wall, legend border, borders to complete axis frame</td>
<td>Color, LineThickness, LineStyle</td>
</tr>
<tr>
<td>GraphCharts</td>
<td>All charts within the graph</td>
<td>Transparency, BackgroundColor, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image, VerticalAlign, TextAlign</td>
</tr>
<tr>
<td>GraphDataText</td>
<td>Text font and color for point and line labels</td>
<td>Font or font-attributes, Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphFloor</td>
<td>3-D floor</td>
<td>BackgroundColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VerticalAlign</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextAlign</td>
</tr>
<tr>
<td>GraphFootnoteText</td>
<td>Text font and color for footnotes</td>
<td>Font or font-attributes&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayopts</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayopts</td>
</tr>
<tr>
<td>GraphLegendBackground</td>
<td>Background color of the legend</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FrameBorder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphOutlines</td>
<td>Outline properties for fill areas such as bars, pie slices,</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td>and box plots.</td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphTitle1Text</td>
<td>Text font and color for the first title</td>
<td>Font or font-attributes&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphTitleText</td>
<td>Text font and color for titles subsequent to the first title</td>
<td>Font or font-attributes&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphValueText</td>
<td>Text font and color for axis tick values and legend values</td>
<td>Font or font-attributes&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>GraphWalls</td>
<td>Vertical walls bounded by axes</td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
</tbody>
</table>

1 *Font-attributes* can be one of the following: `FONTFAMILY=`, `FONTSIZE=`, `FONTSTYLE=`, `FONTWEIGHT=`.

### Table 20.16  Style Elements Affecting Device-Based Non-Grouped Graphical Data Representation

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Default Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphCutLine</td>
<td>Cutline attributes for a dendogram</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td>GraphFinal</td>
<td>Final data for the waterfall chart. Color applies to filled areas.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextColor</td>
</tr>
<tr>
<td>GraphInitial</td>
<td>Initial data for the waterfall chart. Color applies to filled areas.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextColor</td>
</tr>
<tr>
<td>GraphOther</td>
<td>Other data for the graph. Color applies to filled areas.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphOverflow</td>
<td>Overflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextColor</td>
</tr>
<tr>
<td>GraphUnderflow</td>
<td>Underflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>ThreeColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NeutralColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td>ThreeColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NeutralColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td>TwoColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td>TwoColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
</tbody>
</table>

Table 20.17  Style Elements Affecting Device-Based Grouped Graphical Data Representation

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Default Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphData1</td>
<td>Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>GraphData2</td>
<td>Primitives related to 2nd grouped data items</td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>GraphData3</td>
<td>Primitives related to 3rd grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>GraphData4</td>
<td>Primitives related to 4th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>GraphData5</td>
<td>Primitives related to 5th grouped data items</td>
<td>StartColor, BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphData6</td>
<td>Primitives related to 6th grouped data items</td>
<td>StartColor, BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphData7</td>
<td>Primitives related to 7th grouped data items</td>
<td>StartColor, BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphData8</td>
<td>Primitives related to 8th grouped data items</td>
<td>StartColor, BackGroundImage, ContrastColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Color</td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td>Gradient_Direction</td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td>LineStyle</td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>MarkerSize</td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>StartColor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GraphData9</th>
<th>Primitives related to 9th grouped data items</th>
<th>BackGroundImage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GraphData10</th>
<th>Primitives related to 10th grouped data items</th>
<th>BackGroundImage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GraphData11</th>
<th>Primitives related to 11th grouped data items</th>
<th>BackGroundImage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GraphData12</td>
<td>Primitives related to 12th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
</tbody>
</table>
Overview

Style attributes influence the characteristics of individual cells, tables, documents, graphs, and HTML frames. Style attributes exist within style elements and are specified by the STYLE statement on page 455 or the CLASS statement on page 449. 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 432. For information about using style attributes with ODS Statistical Graphics, see the chapter on controlling the appearance of your graphics in SAS Graph Template Language: User’s Guide.

Style attributes can be supplied by SAS or user-defined. Style attributes can be referenced with a style reference. See “Understanding Style References” on page 440 and “style-reference” on page 894 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 429.

For a list of the values that style attributes can specify, see Chapter 21, “Style Attributes,” on page 827. For a list of style elements that you can specify style attributes in, see Chapter 20, “Style Elements,” on page 797.
Style Attributes Tables

Overview

For usage information about these style attributes, such as aliases, restrictions, and examples, see “Style Attributes Detailed Information” on page 855.

EPUB Destination

The EPUB destination includes the following ODS statements:

- ODS EPUB Statement
- ODS EPUB2 Statement
- ODS EPUB3 Statement

Table 21.1  Tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify whether the border is collapsed or separated</td>
<td>“BORDERCOLLAPSE=COLLAPSE</td>
</tr>
<tr>
<td>Specify the text to show in a data tip for various reporting elements, including tables and table cells.</td>
<td>“FLYOVER=“string””</td>
</tr>
<tr>
<td>Specify an ID for the table or table cell</td>
<td>“HTMLID=“string””</td>
</tr>
<tr>
<td>Specify the minimum number of lines of text that must appear in a paragraph before it is forced to move to another page</td>
<td>“NOBREAKSPACE=ON</td>
</tr>
<tr>
<td>Specify the number of lines of text that must appear at the top of a page if a paragraph is separated by a page break</td>
<td>“WIDTH=dimension ”</td>
</tr>
</tbody>
</table>

Table 21.2  Documents

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the minimum number of lines of text that must appear in a paragraph</td>
<td>“ORPHAN=integer”</td>
</tr>
</tbody>
</table>
Task | Attribute
---|---
before it is forced to move to another page | 
Specify the number of lines of text that must appear at the top of a page if a paragraph is separated by a page break | “WINDOW=integer”

---

**Excel Destination**

The ODS destination for Excel includes the **ODS EXCEL Statement**.

*Table 21.3  Graphs*

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs. In many cases, italic and slant map to the same font.</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Change the visual presentation of the text</td>
<td>“TEXTDECORATION=presentation–options”</td>
</tr>
<tr>
<td>Specify the number of spaces that the first line of output is indented</td>
<td>“TEXTINDENT=n”</td>
</tr>
</tbody>
</table>

*Table 21.4  Document*

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in table cells and HTML documents</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>Specify the string to use for the bullets in the contents file</td>
<td>“LISTSTYLE=bullet-type”</td>
</tr>
<tr>
<td>Specify how to handle space characters in table cells</td>
<td>“NOBREAKSPACE=ON</td>
</tr>
</tbody>
</table>
Specify how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted in table cells

“PROTECTSPECIALCHARS=ON | OFF | AUTO”

### Table 21.5  Tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in table cells and HTML documents</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUND=COLOR=color”</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUND=IMAGE=&quot;string&quot;&quot;</td>
</tr>
<tr>
<td>Specify the color of the bottom border of the table</td>
<td>“BORDERBOTTOMCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the bottom border of the selected cell</td>
<td>“BORDERBOTTOMSTYLE=LINE-STYLE”</td>
</tr>
<tr>
<td>Specify the width of the bottom border of a table or table cell</td>
<td>“BORDERBOTTOMWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the border in a table or table cell if the border is just one color</td>
<td>“BORDERCOLOR=color”</td>
</tr>
<tr>
<td>Specify the color of the left border of a table</td>
<td>“BORDERLEFTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the left border of the specified table cell</td>
<td>“BORDERLEFTSTYLE=LINE-STYLE”</td>
</tr>
<tr>
<td>Specify the width of the left border of the table</td>
<td>“BORDERLEFTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the right border of the table</td>
<td>“BORDERRIGHTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the right border of the selected cell</td>
<td>“BORDERRIGHTSTYLE=LINE-STYLE”</td>
</tr>
<tr>
<td>Specify the width of the right border of the table</td>
<td>“BORDERRIGHTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the border style of one or more borders</td>
<td>“BORDERSTYLE=LINE-STYLE-1 &lt;, ..., LINE-STYLE-4 &gt;”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify the color of the top border of a table or table cell</td>
<td>“BORDERTOPCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the top border of the specified table cell</td>
<td>“BORDERTOPSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the top border of the table</td>
<td>“BORDERTOPWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the width of the border of the table</td>
<td>“BORDERWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the text to show in a data tip for table cells</td>
<td>“FLYOVER=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;”</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs.</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify text to place after the table cell or table</td>
<td>“POSTTEXT=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify text to place before the table cell or table</td>
<td>“PRETEXT=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify how less-than signs (&lt;), greater-than signs (&gt;), and ampersands (&amp;) are interpreted in table cells</td>
<td>“PROTECTSPECIALCHARS=ON</td>
</tr>
<tr>
<td>Specify how to handle space characters in table cells</td>
<td>“NOBREAKSPACE=ON</td>
</tr>
<tr>
<td>Specify a string which contains one or more of the following: format, formula, mergeacross, rotate, hidden, wrap</td>
<td>“TAGATTR=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Change the visual presentation of the text</td>
<td>“TEXTDECORATION=presentation–options”</td>
</tr>
</tbody>
</table>
### Task Attribute

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the number of spaces that the first line of output is indented</td>
<td>&quot;TEXTINDENT=n&quot;</td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>&quot;VERTICALALIGN=BOTTOM</td>
</tr>
<tr>
<td>Specify a URL to link to from various reporting elements, including table cells</td>
<td>&quot;URL=&quot;uniform-resource-locator&quot;&quot;</td>
</tr>
<tr>
<td>Specify the width of a table cell, table, line, or a graph</td>
<td>&quot;WIDTH=dimension &quot;</td>
</tr>
</tbody>
</table>

### HTML5 Destination

HTML5 belongs to the markup family of destinations. The following table contains all the style attributes for the markup family and the style attributes that are specific to ODS HTML5.

*Table 21.6  Output and Text*

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>&quot;BACKGROUNDPOSITION=position&quot;</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>&quot;BACKGROUNDIMAGE=&quot;string&quot;&quot;</td>
</tr>
<tr>
<td>Specify whether an image is repeated horizontally, vertically, both, or not repeated</td>
<td>&quot;BACKGROUNDREPEAT=option&quot;&quot;</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>&quot;FONTWEIGHT=weight&quot;</td>
</tr>
<tr>
<td>Specify the height of a table cell, graph, or graphics in an HTML document</td>
<td>&quot;HEIGHT=dimension &quot;</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>&quot;TEXTALIGN=alignment&quot;</td>
</tr>
</tbody>
</table>
### Table 21.7  HTML Document

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>“BACKGROUNDPOSITION=position”</td>
</tr>
<tr>
<td>Specify whether to put a scroll bar in the frame that references the body file</td>
<td>“BODYSCROLLBAR=YES</td>
</tr>
<tr>
<td>Specify the width of the frame that displays the body file in the HTML frame file</td>
<td>“BODYSIZE=dimension</td>
</tr>
<tr>
<td>Specify whether to put a scroll bar in the frames in the frame file that display the contents and the page files</td>
<td>“CONTENTSCROLLBAR=YES</td>
</tr>
<tr>
<td>Specify whether to put a border around the frame for an HTML file that uses frames</td>
<td>“FRAMEBORDER=ON</td>
</tr>
<tr>
<td>Specify the height of a table cell, graph, or graphics in an HTML document</td>
<td>“HEIGHT=dimension ”</td>
</tr>
<tr>
<td>Specify whether to make the entry in the table of contents a link to the body file</td>
<td>“LISTENTRYANCHOR=ON</td>
</tr>
<tr>
<td>Specify a path or URL to an image file to be used as the bullet for list items.</td>
<td>“LISTSTYLEIMAGE=string”</td>
</tr>
<tr>
<td>Specify the bottom margin for the HTML document</td>
<td>“MARGINBOTTOM=dimension”</td>
</tr>
<tr>
<td>Specify the left margin for the HTML document</td>
<td>“MARGINLEFT=dimension”</td>
</tr>
<tr>
<td>Specify the right margin for the HTML document</td>
<td>“MARGINRIGHT=dimension”</td>
</tr>
<tr>
<td>Specify the top margin for the HTML document</td>
<td>“MARGINTOP=dimension”</td>
</tr>
<tr>
<td>Specify the color of the link-focus indicator outline</td>
<td>“OUTLINECOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the link-focus indicator outline</td>
<td>“OUTLINESTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the line width of the link-focus indicator outline</td>
<td>“OUTLINEWIDTH=dimension”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify HTML to place at page breaks in an HTML document</td>
<td>“PAGEBREAKHTML=&quot;string!&quot;”</td>
</tr>
<tr>
<td>Specify the width of a table cell, table, line, or a graph</td>
<td>“WIDTH=dimension ”</td>
</tr>
</tbody>
</table>

**Table 21.8  Tables**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in table cells</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>and HTML documents</td>
<td></td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=&quot;string!&quot;”</td>
</tr>
<tr>
<td>Specify the position of the background of the tables, table cells,</td>
<td>“BACKGROUNDPOSITION=position”</td>
</tr>
<tr>
<td>or graphs</td>
<td></td>
</tr>
<tr>
<td>Specify whether an image is repeated horizontally, vertically, both</td>
<td>“BACKGROUNDREPEAT=option”</td>
</tr>
<tr>
<td>or not repeated</td>
<td></td>
</tr>
<tr>
<td>Specify the color of the bottom border of the table</td>
<td>“BORDERBOTTOMCOLOR=color ”</td>
</tr>
<tr>
<td>Specify the line style of the bottom border of the selected cell</td>
<td>“BORDERBOTTOMSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the bottom border of the table</td>
<td>“BORDERBOTTOMWIDTH=dimension ”</td>
</tr>
<tr>
<td>Specify whether the border is collapsed or separated</td>
<td>“BORDERCOLLAPSE=COLLAPSE</td>
</tr>
<tr>
<td>Specify the color of the border in a table or table cell if the</td>
<td>“BORDERCOLOR=color”</td>
</tr>
<tr>
<td>border is just one color</td>
<td></td>
</tr>
<tr>
<td>Specify the lighter color to use in a border that uses two colors</td>
<td>“BORDERCOLORLIGHT=color”</td>
</tr>
<tr>
<td>to create a three-dimensional effect</td>
<td></td>
</tr>
<tr>
<td>Specify the color of the left border of a table</td>
<td>“BORDERLEFTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the left border of the specified table</td>
<td>“BORDERLEFTSTYLE=line-style”</td>
</tr>
<tr>
<td>cell</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify the width of the left border of the table</td>
<td>“BORDERLEFTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the right border of the table</td>
<td>“BORDERRIGHTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the right border of the selected cell</td>
<td>“BORDERRIGHTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the right border of the table</td>
<td>“BORDERRIGHTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the thickness of the spacing between cells in a table</td>
<td>“BORDERSPACING=dimension”</td>
</tr>
<tr>
<td>Specify the border style of one or more borders</td>
<td>“BORDERSTYLE=line-style-1 &lt;, ..., line-style-4 &gt;”</td>
</tr>
<tr>
<td>Specify the color of the top border of the table</td>
<td>“BORDERTOPCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the top border of the specified table cell</td>
<td>“BORDERTOPSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the top border of the table</td>
<td>“BORDERTOPWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the width of the border of the table</td>
<td>“BORDERWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the amount of white space on each of the four sides of the content in a table cell</td>
<td>“CELLPADDING=dimension</td>
</tr>
<tr>
<td>Specify whether the is collapsed or separated</td>
<td>“BORDERCOLLAPSE=COLLAPSE</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;”</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify the text to show in a data tip for various reporting elements, including tables and table cells.</td>
<td>“FLYOVER=”string”“</td>
</tr>
<tr>
<td>Specify the height of a table cell, graph, or graphics in an HTML document</td>
<td>“HEIGHT=dimension ”</td>
</tr>
<tr>
<td>Specify the window or frame in which to open the target of the link</td>
<td>“HREFTARGET=”target” “</td>
</tr>
<tr>
<td>Specify individual attributes and values for a table or table cell in an HTML document</td>
<td>“HTMLSTYLE=”string”“</td>
</tr>
<tr>
<td>Specify the color of the left of a table</td>
<td>“BORDERLEFTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the amount of white space between the content of the table cell and the border</td>
<td>“PADDING=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the bottom of the content of the table cell</td>
<td>“PADDINGBOTTOM=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the left side of the content of the table cell</td>
<td>“PADDINGLEFT=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the right side of the content of the table cell</td>
<td>“PADDINGRIGHT=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the top of the content of the table cell</td>
<td>“PADDINGTOP=dimension</td>
</tr>
<tr>
<td>Specify the HTML code to place after the table or table cell</td>
<td>“POSTHTML=”string”“</td>
</tr>
<tr>
<td>Specify an image to place before the table or table cell</td>
<td>“POSTIMAGE=”external-file”</td>
</tr>
<tr>
<td>Specify text to place after the table cell or table</td>
<td>“POSTTEXT=”string”“</td>
</tr>
<tr>
<td>Specify the HTML code to place before the table or table cell</td>
<td>“PREHTML=”string”“</td>
</tr>
<tr>
<td>Specify an image to place before the table or table cell</td>
<td>“PREIMAGE=”external-file”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify text to place before the table cell or table</td>
<td>“PRETEXT=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify how less-than signs (&lt;), greater-than signs (&gt;), and ampersands (&amp;) are interpreted in table cells</td>
<td>“PROTECTSPECIALCHARS=ON</td>
</tr>
<tr>
<td>Specify the color of the right of the table</td>
<td>“BORDERRIGHTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the types of rules to use in tables</td>
<td>“RULES=rule-type”</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Change the visual presentation of the text</td>
<td>“TEXTDECORATION=presentation–options”</td>
</tr>
<tr>
<td>Specify the number of spaces that the first line of output is indented</td>
<td>“TEXTINDENT=n”</td>
</tr>
<tr>
<td>Specify the color of the top of the table</td>
<td>“BORDERTOPCOLOR=color”</td>
</tr>
<tr>
<td>Specify a URL to link to from various reporting elements, including table cells</td>
<td>“URL=&quot;uniform-resource-locator&quot;”</td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
</tbody>
</table>

### HTML4 Destination

HTML4 belongs to the markup family of destinations. The following table contains all the style attributes for the markup family and the style attributes that are specific to ODS HTML4.

*Table 21.9  HTML Document*

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the color that a link in an HTML document changes to after you click it, but before the browser opens that file</td>
<td>“ACTIVELINKCOLOR=color”</td>
</tr>
<tr>
<td>Specify whether to put a scroll bar in the frame that references the body file</td>
<td>“BODYSCROLLBAR=YES</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify the width of the frame that displays the body file in the HTML frame file</td>
<td>“BODYSIZE=dimension</td>
</tr>
<tr>
<td>Specify whether to put a scroll bar in the frames in the frame file that display the contents and the page files</td>
<td>“CONTENTSCROLLBAR=YES</td>
</tr>
<tr>
<td>Specify the width of the frames in the frame file that display the contents and the page files</td>
<td>“CONTENTSIZE=dimension</td>
</tr>
<tr>
<td>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</td>
<td>“CONTENTTYPE=&quot;string””</td>
</tr>
<tr>
<td>Specify the entire doctype declaration for the HTML document</td>
<td>“DOCTYPE=&quot;string””</td>
</tr>
<tr>
<td>Specify whether to put a border around the frame for an HTML file that uses frames</td>
<td>“FRAMEBORDER=ON</td>
</tr>
<tr>
<td>Specify the color of the link-focus indicator outline</td>
<td>“OUTLINECOLOR=color”</td>
</tr>
<tr>
<td>Specify whether to make the entry in the table of contents a link to the body file</td>
<td>“LISTENTRYANCHOR=ON</td>
</tr>
<tr>
<td>Specify the color for the links in an HTML document that have not yet been visited</td>
<td>“LINKCOLOR=color”</td>
</tr>
<tr>
<td>Specify HTML to place at page breaks in an HTML document</td>
<td>“PAGEBREAKHTML=&quot;string””</td>
</tr>
<tr>
<td>Specify the color for links that have been visited in an HTML document</td>
<td>“VISITEDLINKCOLOR=color”</td>
</tr>
<tr>
<td>Specify whether to make the image that is specified by BACKGROUNDIMAGE= into a &quot;watermark &quot;</td>
<td>“WATERMARK=ON</td>
</tr>
</tbody>
</table>

Table 21.10  Tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the thickness of the spacing between cells in a table</td>
<td>“BORDERSPACING=dimension”</td>
</tr>
</tbody>
</table>
### Style Attributes Tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify whether the border is collapsed or separated</td>
<td>“BORDERCOLLAPSE=COLLAPSE</td>
</tr>
<tr>
<td>Specify the amount of white space on each of the four sides of the content in a table cell</td>
<td>“CELLPADDING=dimension</td>
</tr>
<tr>
<td>Specify an ID for the table or table cell</td>
<td>“HTMLID=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify the amount of white space between the content of the table cell and the border</td>
<td>“PADDING=dimension</td>
</tr>
<tr>
<td>Specify text to insert in the HTML</td>
<td>“TAGATTR=&quot;string&quot;”</td>
</tr>
</tbody>
</table>

### Markup Family of Destinations

The markup family of destinations includes the following ODS statements:

- ODS CSVALL Statement
- ODS HTML3 Statement
- ODS HTML Statement
- ODS CHTML Statement
- ODS PHTML Statement

#### Table 21.11 Document

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify whether styles used in an HTML document are used in CSS style files</td>
<td>“ABSTRACT=ON</td>
</tr>
</tbody>
</table>

#### Table 21.12 Output and Text

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the alternate colors for maps</td>
<td>“CONTRASTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the entire doctype declaration for the HTML document</td>
<td>“DOCTYPE=&quot;string&quot;”</td>
</tr>
<tr>
<td>Place a rule of the specified width into the space around the text (or entire cell if necessary)</td>
<td>“FILLRULEWIDTH=dimension”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>there is no text) in a table where white space would otherwise appear</td>
<td></td>
</tr>
<tr>
<td>Specify the text to show in a data tip for the table cell</td>
<td>“FLYOVER=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>“FONT=font-definition”</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;”</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph</td>
<td>“FONTWIDTH=relative-width”</td>
</tr>
<tr>
<td>Specify the height of a table cell, graph, or graphics in an HTML document 1</td>
<td>“HEIGHT=dimension”</td>
</tr>
<tr>
<td>Specify the image to appear in a graph</td>
<td>“IMAGE=&quot;external-file&quot;</td>
</tr>
<tr>
<td>Specify the start fill color for a graph</td>
<td>“STARTCOLOR=color”</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Specify a transparency level for graphs</td>
<td>“TRANSPARENCY=dimension”</td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
<tr>
<td>Specify the width of a table cell, table, line, or a graph</td>
<td>“WIDTH=dimension”</td>
</tr>
</tbody>
</table>
Table 21.13  Tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in an HTML document</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUNDCOLOR=color”</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=”string””</td>
</tr>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>“BACKGROUNDPOSITION=position”</td>
</tr>
<tr>
<td>Specify whether an image is repeated horizontally, vertically, both, or not repeated</td>
<td>“BACKGROUNDREPEAT=option”</td>
</tr>
<tr>
<td>Specify the color of the border in a table or table cell if the border is just one color</td>
<td>“BORDERCOLOR=color”</td>
</tr>
<tr>
<td>Specify the color of the bottom of the table</td>
<td>“BORDERBOTTOMCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the bottom of the selected cell</td>
<td>“BORDERBOTTOMSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the bottom of the table</td>
<td>“BORDERBOTTOMWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the amount of white space on each of the four sides of the content in a table cell</td>
<td>“CELLPADDING=dimension</td>
</tr>
<tr>
<td>Specify the name of the style sheet class to use in an HTML document for the table or table cell</td>
<td>“CLASS=’string’”</td>
</tr>
<tr>
<td>Specify the darker color to use in a that uses two colors to create a three-dimensional effect</td>
<td>“BORDERCOLORDARK=color”</td>
</tr>
<tr>
<td>Specify the lighter color to use in a that uses two colors to create a three-dimensional effect</td>
<td>“BORDERCOLORLIGHT=color”</td>
</tr>
<tr>
<td>Specify the window or frame in which to open the target of the link</td>
<td>“HREFTARGET=”target””</td>
</tr>
<tr>
<td>Specify an ID for the table or table cell</td>
<td>“HTMLID=’string’”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Specify the color of the left of a table</td>
<td>“BORDERLEFTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the left of the specified table cell</td>
<td>“BORDERLEFTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the left of the table</td>
<td>“BORDERLEFTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the right of the table</td>
<td>“BORDERRIGHTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the right of the specified cell</td>
<td>“BORDERRIGHTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the right of the table</td>
<td>“BORDERRIGHTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the thickness of the spacing between cells in a table</td>
<td>“BORDERSPACING=dimension”</td>
</tr>
<tr>
<td>Specify the style of one or more sides of a table</td>
<td>“BORDERSTYLE=line-style-1 &lt;, ..., line-style-4 &gt;”</td>
</tr>
<tr>
<td>Specify the color of the top of the table</td>
<td>“BORDERTOPCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the top of the specified table cell</td>
<td>“BORDERTOPSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the top of the table</td>
<td>“BORDERTOPWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the width of the table border</td>
<td>“BORDERWIDTH=dimension”</td>
</tr>
</tbody>
</table>

**PowerPoint Destination**

The ODS destination for PowerPoint includes the ODS POWERPOINT Statement.

Table 21.14  Output and Text

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=&quot;string””</td>
</tr>
<tr>
<td>Specify whether an image is repeated horizontally, vertically, both, or not repeated</td>
<td>“BACKGROUNDREPEAT=option”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;”&quot;</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
</tbody>
</table>

**Table 21.15  Document**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in table cells and HTML documents</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>Specify the string to use for the bullets in the contents file</td>
<td>“LISTSTYLETYPE=bullet-type”</td>
</tr>
<tr>
<td>Specify the left margin for the PowerPoint slide</td>
<td>“MARGINLEFT=dimension”</td>
</tr>
<tr>
<td>Specify the right margin for the PowerPoint slide</td>
<td>“MARGINRIGHT=dimension”</td>
</tr>
</tbody>
</table>

**Table 21.16  Tables**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUNDCOLOR=color”</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify whether an image is repeated horizontally, vertically, both, or not repeated</td>
<td>“BACKGROUNDREPEAT=option”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify the color of the bottom border of the table</td>
<td>“BORDERBOTTOMCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the bottom border of the selected cell</td>
<td>“BORDERBOTTOMSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the bottom border of the table</td>
<td>“BORDERBOTTOMWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the border in a table or table cell if the border is just one color</td>
<td>“BORDERCOLOR=color”</td>
</tr>
<tr>
<td>Specify the color of the left border of a table</td>
<td>“BORDERLEFTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the left border of the specified table cell</td>
<td>“BORDERLEFTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the left border of the table</td>
<td>“BORDERLEFTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the right border of the table</td>
<td>“BORDERRIGHTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the right border of the selected cell</td>
<td>“BORDERRIGHTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the right border of the table</td>
<td>“BORDERRIGHTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the top border of the table</td>
<td>“BORDERTOPCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the top border of the specified table cell</td>
<td>“BORDERTOPSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the top border of the table</td>
<td>“BORDERTOPWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the width of the border of the table</td>
<td>“BORDERWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the amount of white space on each of the four sides of the content in a table cell</td>
<td>“CELLPADDING=dimension</td>
</tr>
<tr>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>“COLOR=color”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify the hyperlink to show in a data tip for tables and table cells</td>
<td>“FLYOVER=”string””</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=”string-1&lt;…, string-n&gt;””</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify the type of frame to use on a table</td>
<td>“FRAME=frame-type”</td>
</tr>
<tr>
<td>Specify the amount of white space on the bottom of the content of the table cell</td>
<td>“PADDINGBOTTOM=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the left side of the content of the table cell</td>
<td>“PADDINGLEFT=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the right side of the content of the table cell</td>
<td>“PADDINGRIGHT=dimension</td>
</tr>
<tr>
<td>Specify the amount of white space on the top of the content of the table cell</td>
<td>“PADDINGTOP=dimension</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Change the visual presentation of the text</td>
<td>“TEXTDECORATION=presentation–options”</td>
</tr>
<tr>
<td>Specify the number of spaces that the first line of output is indented</td>
<td>“TEXTINDENT=n”</td>
</tr>
<tr>
<td>Specify a URL to link to from various reporting elements, including table cells in individual cells</td>
<td>“URL=&quot;uniform-resource-locator””</td>
</tr>
<tr>
<td>Specify vertical justification in individual cells</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
</tbody>
</table>
Printer Family of Destinations

The printer family of destinations includes the following ODS statements:

- ODS PRINTER Statement
- ODS PS Statement
- ODS PCL Statement
- ODS PDF Statement

### Table 21.17  Output and Text

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUNDCOLOR=color”</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=“string””</td>
</tr>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>“BACKGROUNDPOSITION=position”</td>
</tr>
<tr>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>“COLOR=color”</td>
</tr>
<tr>
<td>Specify the alternate colors for maps</td>
<td>“CONTRASTCOLOR=color”</td>
</tr>
<tr>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>“FONT=font-definition”</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;””</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph</td>
<td>“FONTWIDTH=relative-width”</td>
</tr>
</tbody>
</table>

846  Chapter 21 / Style Attributes
<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the height of a table cell, graph, or graphics in an HTML document</td>
<td>“HEIGHT=dimension”</td>
</tr>
<tr>
<td>Specify the image to appear in a graph</td>
<td>“IMAGE=&quot;external-file&quot;</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
</tbody>
</table>

**Table 21.18  Document**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
<td>“FILLRULEWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color for the links in an HTML document that have not yet been visited</td>
<td>“LINKCOLOR=color”</td>
</tr>
<tr>
<td>Specify the string to use for the bullets in the contents file</td>
<td>“LISTSTYLETYPE=bullet-type”</td>
</tr>
<tr>
<td>Specify the bottom margin for the HTML document</td>
<td>“MARGINBOTTOM=dimension”</td>
</tr>
<tr>
<td>Specify the left margin for the HTML document</td>
<td>“MARGINLEFT=dimension”</td>
</tr>
<tr>
<td>Specify the right margin for the HTML document</td>
<td>“MARGINRIGHT=dimension”</td>
</tr>
<tr>
<td>Specify the top margin for the HTML document</td>
<td>“MARGINTOP=dimension”</td>
</tr>
</tbody>
</table>

**Table 21.19  Tables**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in an HTML document</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUNDCOLOR=color”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>&quot;BACKGROUNDIMAGE=&quot;string&quot;&quot;</td>
</tr>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>&quot;BACKGROUNDPOSITION=position&quot;</td>
</tr>
<tr>
<td>Specify the color of the bottom of the table</td>
<td>&quot;BORDERBOTTOMCOLOR=color &quot;</td>
</tr>
<tr>
<td>Specify the line style of the bottom of the selected cell</td>
<td>&quot;BORDERBOTTOMSTYLE=line-style&quot;</td>
</tr>
<tr>
<td>Specify the width of the bottom of the table</td>
<td>&quot;BORDERBOTTOMWIDTH=dimension &quot;</td>
</tr>
<tr>
<td>Specify the amount of white space on each of the four sides of the content in a table cell</td>
<td>&quot;CELLPADDING=dimension</td>
</tr>
<tr>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>&quot;COLOR=color&quot;</td>
</tr>
<tr>
<td>Specify the color of the border in a table or table cell if the border is just one color</td>
<td>&quot;BORDERCOLOR=color&quot;</td>
</tr>
<tr>
<td>Specify the text to show in a data tip for the table cell</td>
<td>&quot;FLYOVER=&quot;string&quot;&quot;</td>
</tr>
<tr>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>&quot;FONT=font-definition&quot;</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>&quot;FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;&quot;</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>&quot;FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>&quot;FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>&quot;FONTWEIGHT=weight&quot;</td>
</tr>
<tr>
<td>Specify the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph</td>
<td>&quot;FONTWIDTH=relative-width&quot;</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify the type of frame to use on a table</td>
<td>“FRAME=frame-type”</td>
</tr>
<tr>
<td>Specify the height of a table cell, graph,</td>
<td>“HEIGHT=dimension”</td>
</tr>
<tr>
<td>or graphics in an HTML document</td>
<td></td>
</tr>
<tr>
<td>Specify how to handle space characters</td>
<td>“NOBREAKSPACE=ON</td>
</tr>
<tr>
<td>in table cells</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the</td>
<td>“PADDINGBOTTOM=dimension</td>
</tr>
<tr>
<td>bottom of the content of the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the</td>
<td>“PADDINGLEFT=dimension</td>
</tr>
<tr>
<td>left side of the content of the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the</td>
<td>“PADDINGRIGHT=dimension</td>
</tr>
<tr>
<td>right side of the content of the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the</td>
<td>“PADDINTOP=dimension</td>
</tr>
<tr>
<td>top of the content of the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify text to place after the table cell or table</td>
<td>“POSTTEXT=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify an image to place before the table or table cell</td>
<td>“PREIMAGE=&quot;external-file&quot;</td>
</tr>
<tr>
<td>Specify text to place before the table cell or table</td>
<td>“PRETEXT=&quot;string&quot;”</td>
</tr>
<tr>
<td>Specify how less-than signs (&lt;), greater-than signs (&gt;, and</td>
<td>“PROTECTSPECIALCHARS=ON</td>
</tr>
<tr>
<td>ampersands (&amp;) are interpreted in table cells</td>
<td></td>
</tr>
<tr>
<td>Specify the width of the right of the table</td>
<td>“BORDERRIGHTWIDTH=dimension ”</td>
</tr>
<tr>
<td>Specify the types of rules to use in tables</td>
<td>“RULES=rule-type”</td>
</tr>
<tr>
<td>Specify the thickness of the spacing</td>
<td>“BORDERSPACING=dimension”</td>
</tr>
<tr>
<td>between cells in a table</td>
<td></td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Change the visual presentation of the text</td>
<td>“TEXTDECORATION=presentation–options”</td>
</tr>
<tr>
<td>Specify the number of spaces that the first line of output is</td>
<td>“TEXTINDENT=n”</td>
</tr>
<tr>
<td>indented</td>
<td></td>
</tr>
</tbody>
</table>
**RTF Family of Destinations**

The RTF family of destinations includes the following statements:

- ODS RTF Statement
- ODS TAGSETS.RTF Statement

**Table 21.20**  Output and Text

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUNDCOLOR=color”</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=&quot;string&quot;&quot;</td>
</tr>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>“BACKGROUNDPOSITION=position”</td>
</tr>
<tr>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>“COLOR=color”</td>
</tr>
<tr>
<td>Specify the alternate colors for maps</td>
<td>“CONTRASTCOLOR=color”</td>
</tr>
<tr>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>“FONT=font-definition”</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;”</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph</td>
<td>“FONTWIDTH=relative-width”</td>
</tr>
<tr>
<td>Specify the image to appear in a graph</td>
<td>“IMAGE=&quot;external-file&quot;</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Specify how to evenly distribute text</td>
<td>“TEXTJUSTIFY=INTER_WORD</td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
</tbody>
</table>

**Table 21.21 Document**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the color for the links in an HTML document that have not yet been visited</td>
<td>“LINKCOLOR=color”</td>
</tr>
<tr>
<td>Specify the bottom margin for the HTML document</td>
<td>“MARGINBOTTOM=dimension”</td>
</tr>
<tr>
<td>Specify the left margin for the HTML document</td>
<td>“MARGINLEFT=dimension”</td>
</tr>
<tr>
<td>Specify the right margin for the HTML document</td>
<td>“MARGINRIGHT=dimension”</td>
</tr>
<tr>
<td>Specify the top margin for the HTML document</td>
<td>“MARGINTOP=dimension”</td>
</tr>
</tbody>
</table>

**Table 21.22 Tables**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how to handle leading spaces and line breaks in an HTML document</td>
<td>“ASIS=ON</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>“BACKGROUNDCOLOR=color”</td>
</tr>
<tr>
<td>Specify an image to use as the background</td>
<td>“BACKGROUNDIMAGE=“string””</td>
</tr>
<tr>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>“BACKGROUNDPOSITION=position”</td>
</tr>
<tr>
<td>Specify the color of the bottom border of the table</td>
<td>“BORDERBOTTOMCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the bottom border of the selected cell</td>
<td>“BORDERBOTTOMSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the bottom border of the table</td>
<td>“BORDERBOTTOMWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify whether the border is collapsed or separated</td>
<td>“BORDER-collapse=COLLAPSE</td>
</tr>
<tr>
<td>Specify the color of the border in a table or table cell if the border is just one color</td>
<td>“BORDERCOLOR=color”</td>
</tr>
<tr>
<td>Specify the color of the left border of a table</td>
<td>“BORDERLEFTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the left border of the specified table cell</td>
<td>“BORDERLEFTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the left border of the table</td>
<td>“BORDERLEFTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the color of the right border of the table</td>
<td>“BORDERRIGHTCOLOR=color”</td>
</tr>
<tr>
<td>Specify the line style of the right border of the selected cell</td>
<td>“BORDERRIGHTSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the right border of the table</td>
<td>“BORDERRIGHTWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the thickness of the spacing between cells in a table</td>
<td>“BORDERSPACING=dimension”</td>
</tr>
<tr>
<td>Specify the border style of one or more borders</td>
<td>“BORDERSTYLE=line-style-1 &lt;, ..., line-style-4 &gt;”</td>
</tr>
<tr>
<td>Specify the color of the top border of the table</td>
<td>“BORDERTOPCOLOR=color”</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify the line style of the top border of the specified table cell</td>
<td>“BORDERTOPSTYLE=line-style”</td>
</tr>
<tr>
<td>Specify the width of the top border of the table</td>
<td>“BORDERTOPWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the width of the border of the table</td>
<td>“BORDERWIDTH=dimension”</td>
</tr>
<tr>
<td>Specify the amount of white space on each of the four sides of the</td>
<td>“CELLPADDING=dimension</td>
</tr>
<tr>
<td>content in a table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>“COLOR=color”</td>
</tr>
<tr>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>“FONT=font-definition”</td>
</tr>
<tr>
<td>Specify the font to use in table cells and graphs</td>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;”</td>
</tr>
<tr>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>“FONTSIZE=dimension</td>
</tr>
<tr>
<td>Specify the style of the font for tables, table cells, and graphs</td>
<td>“FONTSTYLE=ITALIC</td>
</tr>
<tr>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>“FONTWEIGHT=weight”</td>
</tr>
<tr>
<td>Specify the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph</td>
<td>“FONTWIDTH=relative-width”</td>
</tr>
<tr>
<td>Specify the type of frame to use on a table</td>
<td>“FRAME=frame-type”</td>
</tr>
<tr>
<td>Specify the height of a table cell, graph, or graphics in an HTML</td>
<td>“HEIGHT=dimension ”</td>
</tr>
<tr>
<td>document 1</td>
<td></td>
</tr>
<tr>
<td>Specify how to handle space characters in table cells</td>
<td>“NOBREAKSPACE=ON</td>
</tr>
<tr>
<td>Specify the amount of white space between the content of the table</td>
<td>“PADDING=dimension</td>
</tr>
<tr>
<td>cell and the border</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify the amount of white space on the bottom of the content of</td>
<td>“PADDINGBOTTOM=dimension</td>
</tr>
<tr>
<td>the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the left side of the content</td>
<td>“PADDINGLEFT=dimension</td>
</tr>
<tr>
<td>of the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the right side of the content</td>
<td>“PADDINGRIGHT=dimension</td>
</tr>
<tr>
<td>of the table cell</td>
<td></td>
</tr>
<tr>
<td>Specify the amount of white space on the top of the content of the</td>
<td>“PADDINGTOP=dimension</td>
</tr>
<tr>
<td>table cell</td>
<td></td>
</tr>
<tr>
<td>Specify text to place after the table cell or table</td>
<td>“POSTTEXT=”string”</td>
</tr>
<tr>
<td>Specify an image to place before the table or table cell</td>
<td>“PREIMAGE=external-file</td>
</tr>
<tr>
<td>Specify text to place before the table cell or table</td>
<td>“PRETEXT=”string”</td>
</tr>
<tr>
<td>Specify how less-than signs (&lt;), greater-than signs (&gt;), and</td>
<td>“PROTECTSPECIALCHARS=ON</td>
</tr>
<tr>
<td>ampersands (&amp;) are interpreted in table cells</td>
<td></td>
</tr>
<tr>
<td>Specify the types of rules to use in tables</td>
<td>“RULES=rule-type”</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
<tr>
<td>Change the visual presentation of the text</td>
<td>“TEXTDECORATION=presentation–options”</td>
</tr>
<tr>
<td>Specify the number of spaces that the first line of output is</td>
<td>“TEXTINDENT=n”</td>
</tr>
<tr>
<td>indented</td>
<td></td>
</tr>
<tr>
<td>Specify a URL to link to from various reporting elements, including</td>
<td>“URL=”uniform-resource-locator”</td>
</tr>
<tr>
<td>table cells</td>
<td></td>
</tr>
<tr>
<td>Specify vertical justification</td>
<td>“VERTICALALIGN=BOTTOM</td>
</tr>
<tr>
<td>Specify the width of a table cell, table, line, or a graph</td>
<td>“WIDTH=dimension ”</td>
</tr>
</tbody>
</table>
Word Destination (Preproduction)

The ODS destination for Word includes the ODS WORD Statement.

Table 21.23  Document

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the type of bullet to use for lists and the contents file. ODS uses bullets in the contents file.</td>
<td>LISTSTYLETYPE=</td>
</tr>
</tbody>
</table>

Table 21.24  Output and Text

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify that the words are evenly distributed across the page</td>
<td>TEXTJUSTIFY=INTER_WORD</td>
</tr>
<tr>
<td>Specify justification in tables, table cells, and graphs</td>
<td>“TEXTALIGN=alignment”</td>
</tr>
</tbody>
</table>

Table 21.25  Tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the hyperlink to show in a data tip for tables and table cells</td>
<td>“FLYOVER=”string“”</td>
</tr>
</tbody>
</table>

Style Attributes Detailed Information

To quickly locate style attributes that are valid in your destination, see the Style Attributes Tables on page 828.

**ABSTRACT=ON | OFF**

specifies whether styles used in an HTML document are used in CSS style files.

- **ON**
  - specifies that styles are not used in CSS style files.

- **OFF**
  - specifies that styles are used in CSS style files.
Valid in Markup family destinations, ODS HTML5 destination, and the ODS EPUB destination

**ACTIVE LINKCOLOR**=color

specifies the color that a link in an HTML document changes to after you click it, but before the browser opens that file.

Valid in Markup family destinations

See color style attribute value on page 890

**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.

Valid in Markup family, PowerPoint, and Excel

Default OFF

Tip Specify the ASIS= style attribute within the DATA style element to prevent the compressing of blank characters in table cells.

**BACKGROUNDCOLOR**=color

specifies the color of the background of the tables, table cells, or graphs.

Valid in Markup family, printer family, Excel, PowerPoint, and RTF destinations

Alias BACKGROUND=

Interaction The CBACK= option in the SAS/GRAPH GOPTIONS statement overrides the BACKGROUNDCOLOR= attribute.

Tip Generally, the background color of the table cell overrides the background color of the table. You see the background color for the table only as the space between table cells (see “BORDERSPACING=dimension” on page 862).

See color style attribute value on page 890

Examples “Example 1: Creating a Stand-Alone Style” on page 459

“Example 3: Modifying the Default Style with the CLASS Statement” on page 474

**BACKGROUNDIMAGE**="string"

specifies an image in a table, 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, JPEG, or PNG file. Use a simple filename, 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.

**Valid in**  Markup family, Excel, printer family, TAGSETS.RTF, and PowerPoint destinations

**Interactions**  The BACKGROUNDIMAGE= attribute is overridden by the IBACK= and IMAGESTYLE=FIT options in the SAS/GRAPH GOPTIONS statement.

When you apply BACKGROUNDIMAGE= to a cell using the Report Writing Interface, the image adjusts to the size of the cell.

**See**  string attribute value on page 895

**BACKGROUNDPOSITION=** position

specifies the position of the background of the tables, table cells, or graphs.

**position** can be one of the following:

- **BOTTOM**
- **BOTTOM_CENTER**
- **BOTTOM_LEFT**
- **BOTTOM_RIGHT**
- **CENTER**
- **CENTER_BOTTOM**
- **CENTER_CENTER**
- **CENTER_LEFT**
- **CENTER_RIGHT**
- **CENTER_TOP**
- **LEFT**
- **LEFT_BOTTOM**
- **LEFT_CENTER**
- **LEFT_TOP**
- **RIGHT**
- **RIGHT_BOTTOM**
- **RIGHT_CENTER**
- **RIGHT_TOP**
- **TOP**
- **TOP_CENTER**
- **TOP_LEFT**
- **TOP_RIGHT**

**Valid in**  Markup family, printer family, and RTF destinations
BACKGROUNDREPEAT=\textit{option}  
specifies whether an image is repeated horizontally, vertically, both, or not repeated. \textit{option} can be one of the following:

- **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.

Valid in Markup family and PowerPoint destinations  
Restriction The BACKGROUNDREPEAT= attribute is valid in most markup family destinations.

BODYSCROLLBAR=\texttt{YES} | \texttt{NO} | \texttt{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.

Valid in Markup family destinations  
Tip Typically, BODYSCROLLBAR= is set to AUTO.

BODYSIZE=\textit{dimension} | \textit{dimension}\% | *  
specifies the width of the frame that displays the body file in the HTML frame file.

- \textit{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.

Valid in Markup family destinations  
Tip If \textit{dimension} is a nonnegative number, then the unit of measure is pixels.

See \texttt{dimension attribute value on page 892}  
For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination " in SAS Output Delivery System: User’s Guide.
**BORDERBOTTOMCOLOR=**<em>color</em>
specifies the color of the bottom border of a table or table cell.

- **Valid in**: Markup family, printer family, PowerPoint, Excel, RTF, and Measured RTF destinations
- **Tip**: You might also need to specify a **BORDERBOTTOMWIDTH=** attribute to override the style in the ODS destination.
- **See**: color style attribute value on page 890

**BORDERBOTTOMSTYLE=**<em>line-style</em>
specifies the line style of the bottom border of the specified table cell.

- **line-style**
  - can be one of the following:
    - DASHED
    - DOTTED
    - DOUBLE
    - GROOVE
    - HIDDEN
    - INSET
    - OUTSET
    - RIDGE
    - SOLID
- **Valid in**: Markup family, PowerPoint, Excel, RTF, and Measured RTF destinations
- **Tip**: You might also need to specify the **BORDERBOTTOMWIDTH=** attribute to override the style in the ODS destination.

**BORDERBOTTOMWIDTH=**<em>dimension</em>
specifies the width of the bottom border of a table or table cell.

For the ODS destination for Excel, when you specify the following dimensions, the thickness of the border appears as follows:

- Less than 3pt: Border width is thin.
- 3pt, but less than 5pt: Border width is medium.
- 5pt or more: Border width is thick.

- **Valid in**: Markup family, Excel, PowerPoint, printer family, RTF, and Measured RTF destinations
- **See**: dimension attribute value on page 892

**BORDERCOLLAPSE=**<em>COLLAPSE | SEPARATE</em>
specifies whether the border is collapsed or separated.
BORDERCOLOR=color
specifies the border color of a table or table cell. The color is applied to all four borders.

Valid in Markup family, Excel, PowerPoint, printer family, RTF, and Measured RTF destinations

See color style attribute value on page 890

Example “Example 4: Defining a Table and Graph Style” on page 481

BORDERCOLORDARK=color
in a table or table cell, specifies the darker color to use in a border that uses two colors to create a three-dimensional effect.

Valid in Markup family and printer family destinations

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.

See color style attribute value on page 890

Example “Example 4: Defining a Table and Graph Style” on page 481

BORDERCOLORLIGHT=color
in a table or table cell, specifies the lighter color to use in a border that uses two colors to create a three-dimensional effect.

Valid in Markup family and printer family destinations

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.

See color style attribute value on page 890

Example “Example 4: Defining a Table and Graph Style” on page 481

BORDERLEFTCOLOR=color
specifies the color of the left border of the table.

Valid in Markup family, Excel, PowerPoint, printer family, RTF, and Measured RTF destinations

Tip You might also need to specify the BORDERLEFTWIDTH= attribute to override the style in the ODS destination.

See color style attribute value on page 890

BORDERLEFTSTYLE=line-style
specifies the line style of the left border of the specified table cell.

line-style can be one of the following:
<table>
<thead>
<tr>
<th>Style Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BORDERLEFTWIDTH=dimension</strong></td>
<td>specifies the width of the left border of a table or table cell.</td>
</tr>
<tr>
<td>Valid in</td>
<td>Markup family, Excel, PowerPoint, RTF, and Measured RTF destinations</td>
</tr>
<tr>
<td>Tip</td>
<td>You might also need to specify the BORDERLEFTWIDTH= attribute to override the style in the ODS destination.</td>
</tr>
<tr>
<td><strong>BORDERRIGHTCOLOR=color</strong></td>
<td>specifies the color of the right border of a table or table cell.</td>
</tr>
<tr>
<td>Valid in</td>
<td>Markup family, Excel, PowerPoint, printer family, RTF, and Measured RTF destinations</td>
</tr>
<tr>
<td>Tip</td>
<td>You might also need to specify the BORDERRIGHTWIDTH= attribute to override the style in the ODS destination.</td>
</tr>
<tr>
<td><strong>BORDERRIGHTSTYLE=</strong></td>
<td>specifies the line style of the right border of the selected cell.</td>
</tr>
<tr>
<td>line-style</td>
<td>can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- DASHED</td>
</tr>
<tr>
<td></td>
<td>- DOTTED</td>
</tr>
<tr>
<td></td>
<td>- DOUBLE</td>
</tr>
<tr>
<td></td>
<td>- GROOVE</td>
</tr>
<tr>
<td></td>
<td>- HIDDEN</td>
</tr>
<tr>
<td></td>
<td>- INSET</td>
</tr>
<tr>
<td></td>
<td>- OUTSET</td>
</tr>
<tr>
<td></td>
<td>- RIDGE</td>
</tr>
<tr>
<td></td>
<td>- RIDGE</td>
</tr>
<tr>
<td></td>
<td>- SOLID</td>
</tr>
</tbody>
</table>
Valid in  Markup family, Excel, PowerPoint, RTF, and Measured RTF destinations

Tip  You might also need to specify the BORDERRIGHTWIDTH= attribute to override the style in the ODS destination.

BORDERRIGHTWIDTH=\textit{dimension}  
specifies the width of the right border of the table.

Valid in  Markup family, Excel, PowerPoint, printer family, RTF, and Measured RTF destinations

See  \textit{dimension attribute value on page 892}

BORDERSPACING=\textit{dimension}  
specifies the vertical and horizontal thickness of the spacing between cells in a table.

Valid in  Markup family, RTF, and printer family destinations

Alias  CELLSSpacing=

Default  0

Interaction  If BORDERWIDTH= is nonzero, and if the background color of the table cells contrasts with the background color of the table, then the color of the table cell spacing is determined by the table's background.

See  \textit{dimension attribute value on page 892}

Examples  “Example 1: Creating a Stand-Alone Style” on page 459

“Example 3: Modifying the Default Style with the CLASS Statement” on page 474

BORDERSTYLE=\textit{line-style-1}, ..., \textit{line-style-4}  
specifies the border style of one or more borders.

\textit{line-style}  
is one of the following:

- DASHED
- DOTTED
- DOUBLE
- GROOVE
- HIDDEN
- INSET
- NONE
- OUTSET
- RIDGE
- SOLID

The order in which you specify the style determines the style applied to each side.
Table 21.26  BORDERSTYLE= Option Precedence

<table>
<thead>
<tr>
<th>Example</th>
<th>Order</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>borderstyle= double;</td>
<td>All four sides</td>
<td>Double</td>
</tr>
<tr>
<td>borderstyle= solid double;</td>
<td>Top and bottom</td>
<td>Solid</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>Double</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Top</td>
<td>None</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Right and left</td>
<td>Double</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Bottom</td>
<td>Solid</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Top</td>
<td>None</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Right</td>
<td>Double</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Bottom</td>
<td>Solid</td>
</tr>
<tr>
<td>borderstyle= none double</td>
<td>Left</td>
<td>None</td>
</tr>
</tbody>
</table>

Valid in  Markup family, Excel, RTF, and Measured RTF destinations

BORDERTOPCOLOR=\textit{color}

specifies the color of the top border of a table or table cell.

Valid in  Markup family, printer family, RTF, PowerPoint, Excel, and Measured RTF destinations

Tip  Specify the BORDERTOPWIDTH= attribute to override the style in the ODS destination.

See  color style attribute value on page 890

BORDERTOPSTYLE=\textit{line-style}

specifies the line style of the top border of the specified table cell.

\textit{line-style}

can be one of the following:

- DASHED
- DOTTED
- DOUBLE
- GROOVE
- HIDDEN
- INSET
- OUTSET
- RIDGE
- SOLID
Valid in Markup family, Excel, PowerPoint, RTF, and Measured RTF destinations

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.

Tip You might also need to specify the BORDERTOPWIDTH= attribute to override the style in the ODS destination.

**BORDERTOPWIDTH=** *dimension*

specifies the width of the top border of the table or table cell.

Valid in Markup family, Excel, PowerPoint, printer family, RTF, and Measured RTF destinations

See dimension attribute value on page 892

**BORDERWIDTH=** *dimension*

specifies the width of the table borders. The value of BORDERWIDTH= is applied to all four borders.

Valid in Markup family, Excel, PowerPoint, RTF, and printer family destinations

Tip Typically, when BORDERWIDTH=0, the ODS destination sets RULES=NONE (see the discussion about “RULES=rule-type” on page 885) and FRAME=VOID (see the discussion about “FRAME=frame-type” on page 873).

See dimension attribute value on page 892

Examples “Example 1: Creating a Stand-Alone Style” on page 459

“Example 3: Modifying the Default Style with the CLASS Statement” on page 474

**CAPSTYLE=** *line-shape*

specifies the shape of the line at the end of a box whisker. *line-shape* can be one of the following:

- "BRACKET"
- "LINE"
- "NONE"
- "SERIF"

Requirement You must enclose *line-shape* in quotation marks.

**CELLPADDING=** *dimension | dimension%*

specifies the amount of white space on each of the four sides of the content in a table cell.

*dimension* is a nonnegative number or the amount of white space on each of the four sides of the text in a table cell specified as a percentage of the table.
<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family other than HTML5, PowerPoint, printer family, and the RTF destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restrictions</td>
<td>CELLPADDING= is not valid in the HTML5 destination. All padding is done on the table cells.</td>
</tr>
<tr>
<td></td>
<td>In the ODS Destination for PowerPoint, the cellpadding dimension cannot go below 5 pt.</td>
</tr>
<tr>
<td>See</td>
<td>dimension attribute value on page 892</td>
</tr>
<tr>
<td>Example</td>
<td>“Example 3: Modifying the Default Style with the CLASS Statement” on page 474</td>
</tr>
</tbody>
</table>

**CLASS=”string”**

specifies the name of the style sheet class to use in an HTML document for the table or table cell.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>HTMLCLASS=</td>
</tr>
</tbody>
</table>

**COLOR=color**

specifies the color of the foreground in tables, table cells, or graphs, which is primarily the color of text.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family, printer family, PowerPoint, and RTF destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>FOREGROUND=</td>
</tr>
<tr>
<td>Interaction</td>
<td>The COLOR= attribute is overridden by the CBACK= option in the SAS/GRAPH GOPTIONS statement.</td>
</tr>
</tbody>
</table>

**CONNECT=connect-line-type**

specifies the characteristics of a box plot connect line. *connect-line-type* can be one of the following:

- "MAX"
- "MEAN"
- "MEDIAN"
- "MIN"
- "Q1"
- "Q3"

| Requirement | You must enclose connect-line-type in quotation marks. |
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” in *SAS Output Delivery System: User’s Guide*.)

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 that the browser put a scroll bar on the table of contents frame only if the content in that panel is big enough to require scrolling.

Valid in Markup family destinations

Tip Typically, CONTENTSCROLLBAR= is set to AUTO.

See For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” in *SAS Output Delivery System: User’s Guide*.

CONTENT=html-code specifies the HTML code for page breaks.

html-code is a string of html code that defines page breaks.

Valid in HTML5 and EPUB destinations

Example The following example is from Styles.Daisy:

```
content = '<div class="pagebreak" style="text-align: center;
page-break-before: avoid;
page-break-after: always"><span>&#160;</span></div>'
```

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.

Valid in Markup family destinations

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.

See dimension attribute value on page 892
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 attribute value on page 895

Valid in Markup family destinations
Alias HTMLCONTENTTYPE=

CONTRASTCOLOR=\color
specifies the alternate colors for maps. The alternate colors are applied to the
blocks on region areas in block maps.

Valid in Markup family, RTF, and printer family destinations
See color style attribute value on page 890

DATASKIN=CRISP | GLOSS | MATTE | NONE | PRESSED | SHEEN
specifies the type of skin to apply to plots and charts (other than KPIs) to give
them a raised, 3-D appearance.

The DATASKIN= style attribute is valid for the following plots and charts in the
Graph Template Language:

- bar charts
- box plots
- bubble plots
- drop lines
- high-low charts
- histograms
- line charts
- needle plots
- pie charts
- polygon plots
- reference lines
- scatter plots
- series plots
- step plots
vector plots
waterfall charts

The DATASKIN= style attribute is valid for the following plots and charts in the SG procedures:

bar charts
scatter plots
waterfall charts

Table 21.27  DATASKIN Values

<table>
<thead>
<tr>
<th>NONE</th>
<th>CRISP</th>
<th>GLOSS</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MATTE</th>
<th>PRESSED</th>
<th>SHEEN</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Restriction  In the first maintenance release of SAS 9.4 and later releases, the maximum number of skinned graphical elements is limited to 200 per plot in an overlay or prototype layout. When this limit is exceeded for a plot, the specified data skin is not applied to that plot. In that case, use the DATASKINMAX= option in your ODS GRAPHICS statement to increase the maximum limit.

DISPLAYOPTS="display-feature"
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"). "display-feature" can be one of the following:

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.

Requirement You must enclose "display-feature" in quotation marks.

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 attribute value on page 895

Valid in Markup family destinations

Alias HTMLDOCTYPE=

DROP_SHADOW=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 style attribute value on page 890

FILL_PATTERN=fill_pattern_value

specifies the fill pattern to be displayed on the chart. The valid values are: S, E, L1, L2, L3, L4, L5, R1, R2, R3, R4, R5, X1, X2, X3, X4, and X5.
### Restriction
The `FILLPATTERN=` attribute is valid for bar charts only.

### Tip
To display these fill patterns on the bar chart through the style, you must also specify `FILLPATTERN` as one of the `DISPLAYOPTS` in the `GRAPHBAR` style element.

### See
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 808 and "Style Elements Affecting Device-Based Graphics" on page 817.

---

**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.

**Valid in** Printer family destinations

**Tip** If no text is specified, then `FILLRULEWIDTH=` fills the space around the text with hyphen marks. For example: --this-- or this ------.

**See** `dimension attribute value` on page 892

---

**FLYOVER=**"`string`"
specifies the text to show in a data tip for various reporting elements, including table cells.

**string** is the text of the data tip.

**Restriction** When using the ODS PDF destination, the `FLYOVER=` style attribute text should not exceed 1024 chars. If the length exceeds 1024, the text is ignored and a warning message is recorded in the SAS log.

**Requirement** `string` must be enclosed in quotation marks.

**See** `string attribute value` on page 895

**Valid in** Markup family, EPUB, Excel, PowerPoint, Word, and PDF destinations

---

**FONT=**`font-definition`
specifies a font definition to use in tables, table cells, and graphs.

**Valid in** Markup family, RTF, and printer family destinations

**Tips** 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 table cell.

If the system does not recognize the font specified, then it refers to the system's default font. This attribute does not accept concatenated fonts. SAS Graph Styles can specify only one font.

**See** `font-definition attribute value` on page 892

**Example** "Example 3: Modifying the Default Style with the CLASS Statement" on page 474
**FONTFAMILY=**"string-1<..., string-n>"

specifies the font to use in table 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* attribute value on page 895

**Valid in** Markup family, Excel, PowerPoint, RTF, and printer family destinations

**Alias** FONT_FACE=

**Tips** 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 table cell.

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.

**Example** "Example 1: Creating a Stand-Alone Style" on page 459

**FONTSIZE=**`dimension | size`

specifies the size of the font for tables, table 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* attribute value on page 892

*size*

The value of *size* is relative to all other font sizes in the HTML document.

**Range** 1 to 7

**Valid in** Markup family, Excel, PowerPoint, RTF, and printer family destinations

**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 table cell.

**Example** "Example 1: Creating a Stand-Alone Style" on page 459

**FONTSTYLE=**ITALIC | ROMAN | SLANT

specifies the style of the font for tables, table cells, and graphs. In many cases, italic and slant map to the same font.
Valid in  Excel, Markup family, PowerPoint, RTF, and printer family destinations

Alias  FONT_STYLE=

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 table cell.

Examples  “Example 1: Creating a Stand-Alone Style” on page 459

“Example 3: Modifying the Default Style with the CLASS Statement” on page 474

\textbf{FONTWEIGHT=weight}

specifies the font weight of tables, table cells, and graphs. \textit{weight} is any of the following:

\begin{itemize}
  \item MEDIUM
  \item BOLD
  \item DEMI_BOLD
  \item EXTRA_BOLD
  \item LIGHT
  \item DEMI_LIGHT
  \item EXTRA_LIGHT.
\end{itemize}

Valid in  Markup family, Excel, PowerPoint, RTF, and printer family destinations

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.

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 table cell.

Examples  “Example 1: Creating a Stand-Alone Style” on page 459

“Example 1: Creating a Stand-Alone Style” on page 459

\textbf{FONTWIDTH=relative-width}

specifies the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph. \textit{relative-width} is any of the following:

\begin{itemize}
  \item NORMAL
  \item COMPRESSED
  \item EXTRA_COMPRESSED
\end{itemize}
- **NARROW**
- **WIDE**
- **EXPANDED**

Valid in: Markup family, RTF, and printer family destinations

Alias: FONT_WIDTH=

Restriction: Few fonts honor these values.

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 table cell.

Example: “Example 1: Creating a Stand-Alone Style” on page 459

**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>
<thead>
<tr>
<th>Value for frame-type</th>
<th>Frame Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVE</td>
<td>A border at the top</td>
</tr>
<tr>
<td>BELOW</td>
<td>A border at the bottom</td>
</tr>
<tr>
<td>BOX</td>
<td>Borders at the top, bottom, and both sides</td>
</tr>
<tr>
<td>HSIDES</td>
<td>Borders at the top and bottom</td>
</tr>
<tr>
<td>LHS</td>
<td>A border at the left side</td>
</tr>
<tr>
<td>RHS</td>
<td>A border at the right side</td>
</tr>
<tr>
<td>VOID</td>
<td>No borders</td>
</tr>
<tr>
<td>VSIDES</td>
<td>Borders at the left and right sides</td>
</tr>
</tbody>
</table>

Valid in: Markup family, PowerPoint, RTF, and printer family destinations

Example: “Example 3: Modifying the Default Style with the CLASS Statement” on page 474

**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.

Valid in  Markup family destinations

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 table cell, graph, or graphics in an HTML document.

dimension
is a nonnegative number.

See  dimension attribute value on page 892

Valid in  Markup family, RTF, and printer family destinations

Aliases  CELLEIGHT=

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.

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.

Restriction  Use lowercase letters to specify values for HREFTARGET.

_parent
opens the target in the window from which the current window was opened.

Restriction  Use lowercase letters to specify values for HREFTARGET.

_search
opens the target in the browser's search pane.

Restriction  Use lowercase letters to specify values for HREFTARGET.

_self
opens the target in the current window.
**Restriction**  Use lowercase letters to specify values for HREFTARGET.

_**top**_

opens the target in the topmost window.

**Restriction**  Use lowercase letters to specify values for HREFTARGET.

"**name**"

opens the target in the specified window or the frame.

Valid in  Markup family destinations

Default  _self

**Restriction**  Use lowercase letters to specify values for HREFTARGET.

**Requirement**  target must be enclosed in quotation marks.

**HTMLID=**"**string**"

specifies an ID for the table or table cell. The ID is for use by Java Script.

**string**

is the ID text.

**Requirement**  string must be enclosed in quotation marks.

See  string attribute value on page 895

Valid in  Markup family destinations

**HTMLSTYLE=**"**string**"

specifies individual attributes and values for a table or table cell in an HTML document.

**string**

is the name of an attribute or value.

**Requirement**  string must be enclosed in quotation marks.

See  string attribute value on page 895

Valid in  Markup family destinations

**IMAGE=**"**external-file**"

**IMAGE=**"**external-file?DESC=alternative-text**"

specifies the image to appear in a graph. This image is positioned or tiled.

**external-file**

names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

**Requirement**  external-file must be enclosed in quotation marks.

**?DESC=alternative-text**

specifies alternative image text for the image. To change the alternative text for an image, append ?desc=mydesc to the image filename. This option helps to make reports accessible for people with a wide range of abilities. For accessibility, the text should convey the meaning of the image.
If an empty text string is specified, then the image is ignored by screen readers.

Valid in | PDF, EPUB, and HTML5 destinations
---|---
Alias | DESC=

Accessibility note
You can use the ACCESSIBLECHECK system option to write a note to the SAS log if no description is provided for an image.

Example | image="questions.gif?desc=questionmark montage"

Valid in | Markup family, printer family, and RTF destinations
Interaction | The BACK= and IMAGESTYLE=TILE options in the SAS/GRAPH GOPTIONS statement override the IMAGE= attribute.

KPISKIN= | BASIC | MODERN | NONE | ONYX | SATIN
specifies the type of skin to apply to KPI charts to give them a raised, 3-D appearance.

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 its number. pattern-number can be one of the following:

Figure 21.1 Table of Line Patterns

<table>
<thead>
<tr>
<th>Solid</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortDash</td>
<td>2</td>
</tr>
<tr>
<td>MediumDash</td>
<td>4</td>
</tr>
<tr>
<td>LongDash</td>
<td>5</td>
</tr>
<tr>
<td>MediumDashShortDash</td>
<td>8</td>
</tr>
<tr>
<td>DashDashDot</td>
<td>14</td>
</tr>
<tr>
<td>DashDotDot</td>
<td>15</td>
</tr>
<tr>
<td>Dash</td>
<td>20</td>
</tr>
<tr>
<td>LongDashShortDash</td>
<td>26</td>
</tr>
<tr>
<td>Dot</td>
<td>34</td>
</tr>
<tr>
<td>ThinDot</td>
<td>35</td>
</tr>
<tr>
<td>ShortDashDot</td>
<td>41</td>
</tr>
<tr>
<td>MediumDashDotDot</td>
<td>42</td>
</tr>
</tbody>
</table>

LINETHICKNESS= | dimension
specifies the thickness of a line.

See dimension attribute value on page 892

LINKCOLOR= | color
specifies the color for the links in an HTML document that have not yet been visited.

Valid in | Markup family, printer family, and RTF destinations
See | color style attribute value on page 890
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.

Valid in  Markup family destinations

LISTSTYLEIMAGE=string  
specifies a path or URL to an image file to be used as the bullet for list items.

Valid in  Printer family destinations

LISTSTYLETYPE=bullet-type  
specifies the type of bullet to use for lists and the contents file. ODS uses bullets in the contents file.

bullet-type  
The bullet-type value differs for PowerPoint, Markup, Excel, and Word destinations.

The following values are valid in the Markup destinations:

| ARMENIAN | LOWER-GREEK |
| CIRCLE | LOWER-LATIN |
| DECIMAL | LOWER-ROMAN |
| DECIMAL-LEADING-ZERO | NONE |
| DISC | UPPER-ALPHA |
| GEORGIAN | UPPER-LATIN |
| INHERIT | UPPER-ROMAN |
| LOWER-ALPHA | SQUARE |

The following values are valid in the PowerPoint destination:

| ASTERISKS | HYPHEN |
| BOX | LOWER_ROMAN |
| CHECK | PARENTHESESISED_LOWER_LATIN |
| CIRCLE | PARENTHESESISED_DECIMAL |
| DECIMAL | SQUARE |
| DIAMOND | UPPER_ROMAN |
| DISC |

The following values are valid in the Excel destination:

| CIRCLE | SQUARE |
| DISC | NONE |

The following values are valid in the Word destination:

| CIRCLE | LOWER_ROMAN |
| DECIMAL | NONE |
| DECIMAL_LEADINGZERO | SQUARE |
| DISC | UPPER_ALPHA |
MARGINBOTTOM=dimension
specifies the bottom margin for the document.

Valid in Markup family, printer family, and RTF destinations

Alias BOTTOMMARGIN=

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 892

MARGINLEFT=dimension
specifies the left margin for the document.

Valid in Markup family, PowerPoint, printer family, and RTF destinations

Alias LEFTMARGIN=

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 892

MARGINRIGHT=dimension
specifies the right margin for the document.

Valid in Markup family, PowerPoint, printer family, and RTF destinations

Alias RIGHTMARGIN=

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 892

MARGINTOP=dimension
specifies the top margin for the document.
Valid in Markup family, printer family, and RTF destinations

Alias TOPMARGIN=

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 892

MARKERSIZE=dimension specifies the marker size (both width and height).

See dimension attribute value on page 892

MARKERSYMBOL=marker-symbol specifies a marker symbol. marker-symbol can be one of the following:

Figure 21.2 Table of Marker Symbols

<table>
<thead>
<tr>
<th>ArrowDown</th>
<th>Beam</th>
<th>TriangleLeft</th>
<th>HomeDownFilled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asterisk</td>
<td>Plus</td>
<td>TriangleRight</td>
<td>SquareFilled</td>
</tr>
<tr>
<td>Circle</td>
<td>Square</td>
<td>Union</td>
<td>StarFilled</td>
</tr>
<tr>
<td>Diamond</td>
<td>Star</td>
<td>X</td>
<td>TriangleFilled</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Tack</td>
<td>Y</td>
<td>TriangleDownFilled</td>
</tr>
<tr>
<td>LessThan</td>
<td>Tilde</td>
<td>Z</td>
<td>TriangleLeftFilled</td>
</tr>
<tr>
<td>Hash</td>
<td>Triangle</td>
<td>CircleFilled</td>
<td>TriangleRightFilled</td>
</tr>
<tr>
<td>HomeDown</td>
<td>TriangleDown</td>
<td>DiamondFilled</td>
<td></td>
</tr>
</tbody>
</table>

NEUTRALCOLOR=color specifies the middle color in a three-color ramp.

See color style attribute value on page 890

NOBREAKSPACE=ON | OFF specifies how to handle space characters in table 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.

Valid in Markup family, Excel, printer family, and RTF destinations

ORPHAN=integer specifies the minimum number of lines of text that must appear in a paragraph before it is forced to move to another page.

Valid in EPUB destination

Default 2
**OUTLINECOLOR=** `color`

specifies the color of the link-focus indicator outline.

Valid in  HTML5 destination

Note  The OUTLINECOLOR= attribute is valid starting with **SAS 9.4M5**.

Tip  The OUTLINECOLOR= attribute is useful for making your HTML5 documents accessible. See *Enhancing the Appearance of the Link-Focus Indicator* in *Creating Accessible SAS Output Using ODS and ODS Graphics*.

See  color style attribute value on page 890

**OUTLINESTYLE=** `line-style`

specifies the line style of the link-focus indicator outline.

`line-style`

can be one of the following:

- DASHED
- DOTTED
- DOUBLE
- GROOVE
- HIDDEN
- INSET
- OUTSET
- RIDGE
- SOLID

Valid in  HTML5 destination

Note  The OUTLINESTYLE= attribute is valid starting with **SAS 9.4M5**.

Tip  The OUTLINESTYLE= attribute is useful for making your HTML5 documents accessible. See *Enhancing the Appearance of the Link-Focus Indicator* in *Creating Accessible SAS Output Using ODS and ODS Graphics*.

**OUTLINEWIDTH=** `dimension`

specifies the line width of the link-focus indicator outline.

Valid in  HTML5 destination

Note  The OUTLINEWIDTH= attribute is valid starting with **SAS 9.4M5**.

Tip  The OUTLINEWIDTH= attribute is useful for making your HTML5 documents accessible. See *Enhancing the Appearance of the Link-Focus Indicator* in *Creating Accessible SAS Output Using ODS and ODS Graphics*.

See  dimension attribute value on page 892

**OUTPUTHEIGHT=** `dimension`

specifies the height of a graph.
OUTPUTWIDTH=\textit{dimension}

specifies the width of a graph.

\textit{See dimension attribute value on page 892}

PADDING=\textit{dimension} | \textit{dimension}\% 

specifies the amount of white space between the content of the table cell and the border. The value of PADDING= applies to all four sides.

To change the padding of each side, use one or more of the following attributes:

- \textit{PADDINGBOTTOM=} on page 881
- \textit{PADDINGLEFT=} on page 881
- \textit{PADDINGRIGHT=} on page 881
- \textit{PADDINGTOP=} on page 881

valid in Markup family, RTF, and printer family destinations

\textit{See dimension attribute value on page 892}

PADDINGBOTTOM=\textit{dimension} | \textit{dimension}\% 

specifies the amount of white space on the bottom of the content of the table cell.

Valid in Markup family, PowerPoint, RTF, and printer family destinations

Default 0

\textit{See dimension attribute value on page 892}

PADDINGLEFT=\textit{dimension} | \textit{dimension}\% 

specifies the amount of white space on the left side of the content of the table cell.

Valid in Markup family, PowerPoint, RTF, and printer family destinations

Default 0

\textit{See dimension attribute value on page 892}

PADDINGRIGHT=\textit{dimension} | \textit{dimension}\% 

specifies the amount of white space on the right side of the content of the table cell.

Valid in Markup family, PowerPoint, RTF, and printer family destinations

Default 0

\textit{See dimension attribute value on page 892}

PADDINGTOP=\textit{dimension} | \textit{dimension}\% 

specifies the amount of white space on the top of the content of the table cell.

Valid in Markup family, PowerPoint, RTF, and printer family destinations

\textit{See dimension attribute value on page 892}
Default 0

See dimension attribute value on page 892

**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 attribute value on page 895

Valid in Markup family destinations

**POSTHTML="string"**
specifies the HTML code to place after the table or table cell.

*string*

is the HTML code to place after a table or table cell.

Requirement *string* must be enclosed in quotation marks.

See string attribute value on page 895

Valid in Markup family destinations

Example “Example 3: Modifying the Default Style with the CLASS Statement” on page 474

**POSTIMAGE="external-file" | fileref | POSTIMAGE="external-file?DESC=alternative-text"**
specifies an image to place after the table or table cell.

*external-file*

names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

Requirement *external-file* must be enclosed in quotation marks.

*fileref*

is a reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See “Statements” in *SAS DATA Step Statements: Reference* for information about the FILENAME statement.

?DESC=alternative-text

specifies alternative image text for the image. To change the alternative text for an image, append ?desc=mydesc to the image filename. This option helps to make reports accessible for people with a wide range of abilities. For accessibility, the text should convey the meaning of the image.

If an empty text string is specified, then the image is ignored by screen readers.

Valid in PDF, EPUB, and HTML5 destinations

Alias DESC=
### Accessibility note
You can use the `ACCESSIBLECHECK` system option to write a note to the SAS log if no description is provided for an image.

**Example**
```plaintext
postimage="questions.gif?desc=questionmark montage"
```

**Valid in**
Markup family, printer family, PowerPoint, and RTF destinations

**Restriction**
For the ODS destination for PowerPoint, the `POSTIMAGE=` attribute can be used to place an image below a table, but it cannot be used to put an image after a cell.

**POSTTEXT="string"**
specifies text to place after the table cell or table.

**Requirement**
`string` must be enclosed in quotation marks.

**See**
[string attribute value on page 895](#)

**PREHTML="string"**
specifies the HTML code to place before the table or table cell.

**Restriction**
The `PREHTML=` attribute is valid only for markup family destinations.

**See**
[string attribute value on page 895](#)

**PREIMAGE="external-file" | fileref | PREIMAGE="external-file?
DESC=alternative-text"**
specifies an image to place before the table or table cell.

**external-file**
names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

**Valid in**
Markup family, printer family, and RTF destinations

**Restriction**
When using the `PREIMAGE=` style attribute with the PRINTER destination, you must specify `STARTPAGE=NO` on the PRINTER family statement to display page numbers, times, dates, and titles. Without the `STARTPAGE=NO` option, preimages are treated like graphs and have no page numbers, times, dates, or titles displayed.

**Requirement**
Enclose `external-file` in quotation marks.

**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 DATA Step Statements: Reference*.)

**?DESC=alternative-text**
specifies alternative image text for the image. To change the alternative text for an image, append `?desc=mydesc` to the image filename. This option helps to make reports accessible for people with a wide range of abilities. For accessibility, the text should convey the meaning of the image.

If an empty text string is specified, then the image is ignored by screen readers.
### DESC= (Accessibility note)

You can use the **ACCESSIBLECHECK** system option to write a note to the SAS log if no description is provided for an image.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>PDF, EPUB, and HTML5 destinations</th>
</tr>
</thead>
</table>

| Example           | `preimage="questions.gif?desc=question mark montage"` |

### PRETEXT= "string"

Specifies text to place before the table cell or table.

- **string**
  - Text that is placed before the table cell or table.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family, printer family, and RTF destinations</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Enclose <em>string</em> in quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>See</td>
<td><code>string attribute value on page 895</code></td>
</tr>
<tr>
<td>Example</td>
<td>“Customizing the Table of Contents” in <em>SAS Output Delivery System: User’s Guide</em></td>
</tr>
</tbody>
</table>

### PROTECTSPECIALCHARS=ON | OFF | AUTO

Specifies how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted in table 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.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family, Excel, printer family, and RTF destinations</th>
</tr>
</thead>
</table>
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 21.29  RULES= Attribute Values

<table>
<thead>
<tr>
<th>Value of RULES= Attribute</th>
<th>Locations of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Between all rows and columns</td>
</tr>
<tr>
<td>COLS</td>
<td>Between all columns</td>
</tr>
<tr>
<td>GROUPS</td>
<td>Between the table header and the table and between the table and the table footer, if there is one</td>
</tr>
<tr>
<td>NONE</td>
<td>No rules anywhere</td>
</tr>
<tr>
<td>ROWS</td>
<td>Between all rows</td>
</tr>
</tbody>
</table>

Valid in  Markup family, PowerPoint, printer family, and RTF destinations

Example  “Example 4: Defining a Table and Graph Style” on page 481

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 is completely transparent gradationally to the specified start color.

Valid in  HTML4 destination

See  color style attribute value on page 890

TAGATTR="string"
specifies text to insert into HTML.

string is the text that is inserted into HTML tags.

Requirements  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.

Interaction  When using the Report Writing Interface with the ODS destination for Excel, the TAGATTR= Style Attribute is needed
to pass Excel formats instead of the FORMAT parameter within the FORMAT_CELL Method.

See string attribute value on page 895

Valid in Markup family and Excel destinations

**TEXTALIGN=alignment**
specifies justification in tables, table cells, and graphs. In graphs, this option specifies the justification of the image specified with the IMAGE= statement. 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; 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; The value of alignment can be one of the following:

**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.

Valid in Markup family, printer family, PowerPoint, Excel, Word, and RTF destinations

Alias JUST=

Restriction For the HTML5 destination, you might be able to use MARGINRIGHT=0 instead.

Tips 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 printer family destinations and the MARKUP destination, control the placement of dates by using the style attribute TEXTALIGN= with
the style attribute VERTICALALIGN= in the BodyDate or Date. style element.

**TEXTDECORATION=presentation–options**
changes the visual presentation of the text. presentation–options can be one of the following:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Valid in</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLINK</td>
<td>specifies that the text's visual presentation alternates rapidly between visible and invisible.</td>
<td>Excel, HTML, RTF, PowerPoint, and Excel destinations</td>
</tr>
<tr>
<td>LINE_THROUGH</td>
<td>specifies that a line is drawn through the text.</td>
<td>HTML, printer family, measured RTF, and RTF destinations</td>
</tr>
<tr>
<td>OVERLINE</td>
<td>specifies that a line is drawn above the text.</td>
<td>HTML and printer family destinations</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>specifies that a line is drawn below the text.</td>
<td>HTML, printer family, measured RTF, and RTF destinations</td>
</tr>
</tbody>
</table>

Tip: **TEXTDECORATION=** can be used with inline formatting and the ODS PDF statement to enhance PDF files.

Example: "Formatting Cells with the Textdecoration Style Attribute " in SAS Output Delivery System: Advanced Topics

**TEXTINDENT=n**
specifies the number of spaces that the first line of output is indented.

<table>
<thead>
<tr>
<th>n</th>
<th>Specifies the number of spaces to indent the output.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid in</td>
<td>Markup family, printer family, Excel, PowerPoint, and RTF destinations</td>
</tr>
<tr>
<td>Alias</td>
<td>INDENT=</td>
</tr>
<tr>
<td>Default</td>
<td>The default value for XML is 2. For all other ODS destinations, the default value is 0.</td>
</tr>
</tbody>
</table>

**TICKDISPLAY="INSIDE" | OUTSIDE" | ACROSS"**
specifies the placement of all major and minor axis tick marks.

**TEXTJUSTIFY=INTER_WORD | INTER_CHARACTER**
specifies how to evenly distribute text.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Valid in</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTER_WORD</td>
<td>specifies that the words are evenly distributed across the page.</td>
<td>RTF, TAGSETS.RTF, and Word destinations</td>
</tr>
<tr>
<td>INTER_CHARACTER</td>
<td>specifies that all characters are evenly distributed across a page.</td>
<td>RTF, TAGSETS.RTF, and Word destinations</td>
</tr>
</tbody>
</table>
**Restriction**  The ODS destination for Word supports only the INTER_WORD option.

**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).

<table>
<thead>
<tr>
<th>Valid in</th>
<th>HTML destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>See</td>
<td>dimension attribute value on page 892</td>
</tr>
</tbody>
</table>

**URL="uniform-resource-locator"**  specifies a URL to link to from various reporting elements, including table cells.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family, Excel, PowerPoint, printer family, and RTF destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>uniform-resource-locator must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

**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=. 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;  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;

**BOTTOM**  specifies bottom justification.

  Alias  B

**MIDDLE**  specifies center justification.

  Alias  M

**TOP**  specifies top justification.

  Alias  T

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family, Excel, printer family, and RTF destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>VJUST=</td>
</tr>
</tbody>
</table>

**Tips**  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 printer and markup family destinations, control the placement of dates by using the style attribute VERTICALALIGN= with the style attribute TEXTALIGN= in the BODYDATE or DATE style element.
**VISITEDLINKCOLOR=**\textit{color}  
specifies the color for links that have been visited in an HTML document.  

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>See</td>
<td>color style attribute value on page 890</td>
</tr>
</tbody>
</table>

**WATERMARK=**\texttt{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.  

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family and RTF destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>See</td>
<td>“BACKGROUNDIMAGE=&quot;string&quot;” on page 856</td>
</tr>
</tbody>
</table>

**WIDTH=**\textit{dimension}  
specifies the width of a table cell, table, line, or a graph.  

When used with graphs, the WIDTH= option must be specified as a pixel or percentage value. If a unit of measure is not specified with the \textit{dimension}, then the value will be in pixels. If a unit of measure other than pixels or percentage is specified with the \textit{dimension}, then the HEIGHT=\textit{dimension} is not applied to the graph.  

\textit{dimension}  
is a nonnegative number.  

<table>
<thead>
<tr>
<th>Valid in</th>
<th>Markup family, Excel, printer family, and RTF destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>CELLWIDTH=</td>
</tr>
<tr>
<td>Alias</td>
<td>OUTPUTWIDTH=</td>
</tr>
<tr>
<td>Restriction</td>
<td>The WIDTH= option does not apply to output generated as a result of GRSEG (graph segment) output.</td>
</tr>
<tr>
<td>Interaction</td>
<td>The XPIXELS= option in the SAS/GRAPH GOPTIONS statement overrides the WIDTH= attribute.</td>
</tr>
<tr>
<td>Tips</td>
<td>A column of cells has the width of the widest cell in the column.</td>
</tr>
<tr>
<td></td>
<td>Use WIDTH=100% to make the table or graph as wide as the window that it is open in.</td>
</tr>
</tbody>
</table>
**WINDOW=integer**

specifies the number of lines of text that must appear at the top of a page if a paragraph is separated by a page break.

Valid in EPUB destination

Default 2

---

**Style Attributes Values**

**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).
  - a red/green/blue transparency (RGBA) value (for example, a98FB9880)
  - a cyan/magenta/yellow/black (CMYK) value (for example, FFFFFF00)

---

Note: RGBA color mode is not supported by Java devices. RGBA color mode is supported by ActiveX devices when the output is used in Microsoft applications.

---

- an RGB value with a leading number sign (#) rather than CX (for example, #0023FF).
- one of the colors that exist in the SAS session when the style is used:
  - DMSBLUE
  - DMSRED
  - DMSPINK
  - DMSGREEN
  - DMSCYAN
  - DMSYELLOW
  - DMSWHITE
  - DMSORANGE
  - DMSBLACK
  - DSMAGENTA
  - DMSGRAY
  - DMSBROWN
  - SYSBACK
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 21.30 Hue/Light/Saturation (HLS) Values

<table>
<thead>
<tr>
<th>Lightness</th>
<th>Saturation</th>
<th>Chromatic Hue</th>
<th>Achromatic Hue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Blue</td>
<td>Black¹</td>
</tr>
<tr>
<td>Very dark</td>
<td>Grayish</td>
<td>Purple</td>
<td></td>
</tr>
<tr>
<td>Dark</td>
<td>Moderate</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Strong</td>
<td>Orange</td>
<td>brown</td>
</tr>
<tr>
<td>Light</td>
<td>Vivid</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>Very light</td>
<td></td>
<td>Green</td>
<td></td>
</tr>
</tbody>
</table>

³ White¹

1 Black and white cannot be combined with a lightness value or a saturation value.
2 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
- bluish purple
- reddish orange
- yellowish green

**Tips** For a list of some valid colors, see [Link to Valid Colors to use with cascading style sheets](#).

To see how color names map to hexadecimal values, submit the following REGISTRY procedure code:

```
proc registry list startat="COLORNAMES";
run;
```

See [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 21.31 Units of Measure for Dimension**

<table>
<thead>
<tr>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
</tbody>
</table>

**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)
```

A font specification in an ODS style can specify multiple fonts so that a reasonable substitution can be made when a font cannot be located on the
current computer. The fonts are normally listed in a most-specific to most-generic order. For example, the TitleFont attribute shown in the following example specifies a list of fonts in the following order: <sans-serif>, <MTsans-serif>, Helvetica, and then Helv.

"TitleFont"=('<sans-serif>,<MTsans-serif>,Helvetica,Helv',3,bold)

For table title text, SAS searches the computer for a font in the TitleFont font list in the order listed. The first font in the list that is found is used.

If a font is enclosed in angle brackets, that indicates that the resolved font is defined in the ODS\FONTS portion of the SAS registry. One way to view the SAS registry font definitions is with the DMS REGEDIT command.

"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.

Note For SAS 9.4M5, ODS Styles supports the AvenirNextForSAS font.

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 attribute value on page 892.

"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 21.32  Font Keywords

<table>
<thead>
<tr>
<th>Keywords for Font Weight</th>
<th>Keywords for Font Style</th>
<th>Keywords for Font Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIUM</td>
<td>ITALIC</td>
<td>NORMAL¹</td>
</tr>
<tr>
<td>Keywords for Font Weight</td>
<td>Keywords for Font Style</td>
<td>Keywords for Font Width</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>BOLD</td>
<td>ROMAN</td>
<td>COMPRESSED(^1)</td>
</tr>
<tr>
<td>DEMI_BOLD(^1)</td>
<td>SLANT</td>
<td>EXTRA_COMPRESSED(^1)</td>
</tr>
<tr>
<td>EXTRA_BOLD(^1)</td>
<td>OBLIQUE(^1)</td>
<td>NARROW(^1)</td>
</tr>
<tr>
<td>LIGHT</td>
<td>NORMAL(^1)</td>
<td>WIDE(^1)</td>
</tr>
<tr>
<td>DEMI_LIGHT(^1)</td>
<td></td>
<td>EXPANDED(^1)</td>
</tr>
<tr>
<td>EXTRA_LIGHT(^1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NORMAL(^1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) Few fonts honor these values.

Example  "Example 2: Using User-Defined Attributes" on page 466

**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.

**See**
“Understanding Style References” on page 440

**Example**
“Example 2: Using User-Defined Attributes” on page 466

“*string*”
is a quoted character string.

**user-defined-format**
specifies a format created with the FORMAT procedure.

**Restriction**
*user-defined-format* can be specified only for data cells.
Appendices

Appendix 1
Output Object Table Names ................................................. 899

Appendix 2
Lua License .............................................................................. 927
## ODS Table Names Produced by Base SAS Procedures

This table lists the output object table names that 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.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>Calendar</td>
</tr>
</tbody>
</table>
### Table A13.2  ODS Table Names Produced by the CATALOG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog_Random</td>
<td>Table generated when the catalog is in a random-access data library</td>
</tr>
<tr>
<td>Catalog_Sequential</td>
<td>Table generated when the catalog is in a sequential-access data library</td>
</tr>
</tbody>
</table>

### Table A13.3  ODS Table Names Produced by the CHART Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>Block chart</td>
</tr>
<tr>
<td>Hbar</td>
<td>Horizontal bar chart</td>
</tr>
<tr>
<td>Pie</td>
<td>Pie chart</td>
</tr>
<tr>
<td>Star</td>
<td>Star chart</td>
</tr>
<tr>
<td>Vbar</td>
<td>Vertical bar chart</td>
</tr>
</tbody>
</table>

### Table A13.4  ODS Table Names Produced by the COMPARE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareDatasets</td>
<td>Information about the data set or data sets</td>
<td>Omit NOSUMMARY or NOVALUE options</td>
</tr>
<tr>
<td>CompareDetails</td>
<td>List of observations that the base data set and the compare data set do not have in common</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>CompareDifferences</td>
<td>Report of variable value differences</td>
<td>Omit NOVALUES option</td>
</tr>
<tr>
<td>CompareSummary</td>
<td>Summary report of observations, values, and</td>
<td></td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CompareVariables</td>
<td>List of differences in variable types or attributes between the base data set and the compare data set</td>
<td>Omit NOSUMMARY option unless the variables are identical</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the ID Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareDetails</td>
<td>List of notes and warnings concerning duplicate ID variable values if duplicate ID variable values exist in either of the data sets</td>
</tr>
</tbody>
</table>

**Table A13.5  ODS Table Names Produced by the CORR Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cov</td>
<td>Covariances</td>
<td>COV</td>
</tr>
<tr>
<td>CronbachAlpha</td>
<td>Coefficient alpha</td>
<td>ALPHA</td>
</tr>
<tr>
<td>CronbachAlphaDel</td>
<td>Coefficient alpha with deleted variable</td>
<td>ALPHA</td>
</tr>
<tr>
<td>Csscp</td>
<td>Corrected sums of squares and crossproducts</td>
<td>CSSCP</td>
</tr>
<tr>
<td>FisherPearsonCorr</td>
<td>Pearson correlation statistics using Fisher’s z transformation</td>
<td>FISHER</td>
</tr>
<tr>
<td>FisherSpearmanCorr</td>
<td>Spearman correlation statistics using Fisher’s z transformation</td>
<td>FISHER SPEARMAN</td>
</tr>
<tr>
<td>HoeffdingCorr</td>
<td>Hoeffding’s D statistics</td>
<td>HOEFFDING</td>
</tr>
<tr>
<td>KendallCorr</td>
<td>Kendall’s tau-b coefficients</td>
<td>KENDALL</td>
</tr>
<tr>
<td>PearsonCorr</td>
<td>Pearson correlations</td>
<td>PEARSON</td>
</tr>
<tr>
<td>PolychoricCorr</td>
<td>Polychoric correlations</td>
<td>POLYCHORIC</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>PolyserialCorr</td>
<td>Polyserial correlations</td>
<td>POLYSERIAL</td>
</tr>
<tr>
<td>SimpleStats</td>
<td>Simple descriptive statistics</td>
<td></td>
</tr>
<tr>
<td>SpearmanCorr</td>
<td>Spearman correlations</td>
<td>SPEARMAN</td>
</tr>
<tr>
<td>Sscp</td>
<td>Sums of squares and crossproducts</td>
<td>SSCP</td>
</tr>
<tr>
<td>VarInformation</td>
<td>Variable information</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PARTIAL Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FisherPearsonPartialCorr</td>
<td>Pearson partial correlation statistics using Fisher's z transformation</td>
<td>FISHER</td>
</tr>
<tr>
<td>FisherSpearmanPartialCorr</td>
<td>Spearman partial correlation statistics using Fisher's z transformation</td>
<td>FISHER SPEARMAN</td>
</tr>
<tr>
<td>PartialCsscp</td>
<td>Partial corrected sums of squares and crossproducts</td>
<td>CSSCP</td>
</tr>
<tr>
<td>PartialCov</td>
<td>Partial covariances</td>
<td>COV</td>
</tr>
<tr>
<td>PartialKendallCorr</td>
<td>Partial Kendall tau-b coefficients</td>
<td>KENDALL</td>
</tr>
<tr>
<td>PartialPearsonCorr</td>
<td>Partial Pearson correlations</td>
<td></td>
</tr>
<tr>
<td>PartialSpearmanCorr</td>
<td>Partial Spearman correlations</td>
<td>SPEARMAN</td>
</tr>
</tbody>
</table>

*Table A13.6*  
ODS Table Names Produced by the DATASETS and CONTENTS Procedures
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory</td>
<td>General library information</td>
<td>Omit NOLIST option</td>
</tr>
<tr>
<td>Members</td>
<td>Library member information</td>
<td>Omit NOLIST option</td>
</tr>
</tbody>
</table>

Table A13.7 ODS Table Names Produced by the CONTENTS Procedure or the DATASETS Procedure with the CONTENTS Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Data set attributes</td>
<td>Omit SHORT option</td>
</tr>
<tr>
<td>Directory</td>
<td>General library information</td>
<td>DATA=&lt;libref.&gt;<em>ALL</em> or the DIRECTORY option¹</td>
</tr>
<tr>
<td>EngineHost</td>
<td>Engine and operating</td>
<td>Omit SHORT option</td>
</tr>
<tr>
<td></td>
<td>environment information</td>
<td></td>
</tr>
<tr>
<td>IntegrityConstraints</td>
<td>List of integrity constraints</td>
<td>Omit SHORT option and data has integrity constraints</td>
</tr>
<tr>
<td>IntegrityConstraintsShort</td>
<td>Concise listing of integrity constraints</td>
<td>SHORT option specified and data has integrity constraints</td>
</tr>
<tr>
<td>Indexes</td>
<td>List of indexes</td>
<td>Omit SHORT option and data set is indexed</td>
</tr>
<tr>
<td>IndexesShort</td>
<td>Concise list of indexes</td>
<td>SHORT option specified and data set is indexed</td>
</tr>
<tr>
<td>Members</td>
<td>Library member information</td>
<td>DATA=&lt;libref.&gt;<em>ALL</em> or the DIRECTORY option¹</td>
</tr>
<tr>
<td>Position</td>
<td>List of variables by logical position in the data set</td>
<td>Omit SHORT option and specify the VARNUM option</td>
</tr>
<tr>
<td>PositionShort</td>
<td>Concise list of variables by logical position in the data set</td>
<td>SHORT and VARNUM options</td>
</tr>
<tr>
<td>Sortedby</td>
<td>Sort information</td>
<td>Omit SHORT option and data set is sorted</td>
</tr>
</tbody>
</table>
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 A13.8 ODS Table Names Produced by the FREQ Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>SortedbyShort</td>
<td>Concise sort information</td>
<td></td>
<td>SHORT option and data set is sorted</td>
</tr>
<tr>
<td>Variables</td>
<td>List of variables in alphabetical order</td>
<td></td>
<td>Omit SHORT option</td>
</tr>
<tr>
<td>VariablesShort</td>
<td>Concise listing of variables in alphabetical order</td>
<td></td>
<td>SHORT</td>
</tr>
<tr>
<td>BarnardsTest</td>
<td>Barnard's exact test</td>
<td>EXACT</td>
<td>BARNARD</td>
</tr>
<tr>
<td>BinomialCLs</td>
<td>Binomial confidence limits</td>
<td>TABLES</td>
<td>BINOMIAL(AC</td>
</tr>
<tr>
<td>BinomialEquiv</td>
<td>Binomial equivalence analysis</td>
<td>TABLES</td>
<td>BINOMIAL(EQUIV)</td>
</tr>
<tr>
<td>BinomialEquivLimits</td>
<td>Binomial equivalence limits</td>
<td>TABLES</td>
<td>BINOMIAL(EQUIV)</td>
</tr>
<tr>
<td>BinomialEquivTest</td>
<td>Binomial equivalence test</td>
<td>TABLES</td>
<td>BINOMIAL(EQUIV)</td>
</tr>
<tr>
<td>BinomialNoninf</td>
<td>Binomial noninferiority test</td>
<td>TABLES</td>
<td>BINOMIAL(NONINF)</td>
</tr>
<tr>
<td>Binomial</td>
<td>Binomial proportion</td>
<td>TABLES</td>
<td>BINOMIAL</td>
</tr>
<tr>
<td>BinomialTest</td>
<td>Binomial proportion test</td>
<td>TABLES</td>
<td>BINOMIAL</td>
</tr>
<tr>
<td>BinomialSup</td>
<td>Binomial superiority test</td>
<td>TABLES</td>
<td>BINOMIAL(SUP)</td>
</tr>
<tr>
<td>BreslowDayTest</td>
<td>Breslow-Day test</td>
<td>TABLES</td>
<td>CMH (h x 2 x 2 table)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>CMH</td>
<td>Cochran-Mantel-Haenszel statistics</td>
<td>TABLES</td>
<td>CMH</td>
</tr>
<tr>
<td>ChiSq</td>
<td>Chi-square tests</td>
<td>TABLES</td>
<td>CHISQ</td>
</tr>
<tr>
<td>CochransQ</td>
<td>Cochran's</td>
<td>TABLES</td>
<td>AGREE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>ColScores</td>
<td>Column scores</td>
<td>TABLES</td>
<td>SCOROUT</td>
</tr>
<tr>
<td>CommonOddsRatioCl</td>
<td>Exact confidence limits for the common odds ratio</td>
<td>EXACT</td>
<td>COMOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>CommonOddsRatioTest</td>
<td>Common odds ratio exact test</td>
<td>EXACT</td>
<td>COMOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>CommonPdiff</td>
<td>Common proportion difference</td>
<td>TABLES</td>
<td>RISKDIFF(COMMON)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>CommonRelRisks</td>
<td>Common relative risks</td>
<td>TABLES</td>
<td>CMH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>CrossList</td>
<td>Crosstabulation table in column format</td>
<td>TABLES</td>
<td>CROSSLIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(n\text{-way table, } n &gt; 1)$</td>
</tr>
<tr>
<td>CrossTabFreqs</td>
<td>Crosstabulation table</td>
<td>TABLES</td>
<td>$(n\text{-way table, } n &gt; 1)$</td>
</tr>
<tr>
<td>EqualKappaTest</td>
<td>Test for equal simple kappas</td>
<td>TABLES</td>
<td>AGREE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>EqualKappaTests</td>
<td>Tests for equal kappas</td>
<td>TABLES</td>
<td>AGREE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times r \times r, r &gt; 2)$</td>
</tr>
<tr>
<td>EqualOddsRatios</td>
<td>Tests for equal odds ratios</td>
<td>EXACT</td>
<td>EQOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>GailSimon</td>
<td>Gail-Simon test</td>
<td>TABLES</td>
<td>GAILSIMON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(h \times 2 \times 2 \text{ table})$</td>
</tr>
<tr>
<td>FishersExact</td>
<td>Fisher’s exact test</td>
<td>EXACT</td>
<td>FISHER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FISHER or EXACT</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>FishersExactMC</td>
<td>Monte Carlo estimates for Fisher’s exact test</td>
<td>EXACT</td>
<td>FISHER / MC</td>
</tr>
<tr>
<td>Gamma</td>
<td>Gamma</td>
<td>TEST</td>
<td>GAMMA</td>
</tr>
<tr>
<td>GammaTest</td>
<td>Gamma test</td>
<td>TEST</td>
<td>GAMMA</td>
</tr>
<tr>
<td>JTTTest</td>
<td>Jonckheere-Terpstra test</td>
<td>TABLES</td>
<td>JT</td>
</tr>
<tr>
<td>JTTTestMC</td>
<td>Monte Carlo estimates for Jonckheere-Terpstra exact test</td>
<td>EXACT</td>
<td>JT / MC</td>
</tr>
<tr>
<td>Kappa</td>
<td>Simple kappa coefficient</td>
<td>TEST</td>
<td>KAPPA</td>
</tr>
<tr>
<td>KappaMC</td>
<td>Monte Carlo exact test for simple kappa coefficient</td>
<td>EXACT</td>
<td>KAPPA / MC</td>
</tr>
<tr>
<td>KappaStatistics</td>
<td>Kappa statistics</td>
<td>TABLES</td>
<td>AGREE, no TEST or EXACT (r x r table, &gt; 2)</td>
</tr>
<tr>
<td>KappaTest</td>
<td>Simple kappa test</td>
<td>TEST</td>
<td>KAPPA</td>
</tr>
<tr>
<td>KappaWeights</td>
<td>Kappa weights</td>
<td>TABLES</td>
<td>AGREE(PRINTKW TS)</td>
</tr>
<tr>
<td>List</td>
<td>List format multiway table</td>
<td>TABLES</td>
<td>LIST</td>
</tr>
<tr>
<td>LRChiSq</td>
<td>Likelihood ratio chi-square exact test</td>
<td>EXACT</td>
<td>LRCHI</td>
</tr>
<tr>
<td>LRChiSqMC</td>
<td>Monte Carlo exact test for likelihood ratio chi-square</td>
<td>EXACT</td>
<td>LRCHI / MC</td>
</tr>
<tr>
<td>MantelFleiss</td>
<td>Mantel-Fleiss criterion</td>
<td>TABLES</td>
<td>CMH(MF) (h x 2 x 2 table)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------</td>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>McNemarsTest</td>
<td>McNemar's test</td>
<td>TABLES</td>
<td>AGREE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>Measures</td>
<td>Measures of association</td>
<td>TABLES</td>
<td>MEASURES</td>
</tr>
<tr>
<td>MHChiSq</td>
<td>Mantel-Haenszel chi-square exact test</td>
<td>EXACT</td>
<td>MHCHI</td>
</tr>
<tr>
<td>MHChiSqMC</td>
<td>Monte Carlo exact test for Mantel-Haenszel chi-square</td>
<td>EXACT</td>
<td>MHCHI / MC</td>
</tr>
<tr>
<td>NLevels</td>
<td>Number of variable levels</td>
<td>PROC</td>
<td>NLEVELS</td>
</tr>
<tr>
<td>OddsRatioCLs</td>
<td>Odds ratio confidence limits</td>
<td>TABLES</td>
<td>OR(CL=) (2 × 2 table)</td>
</tr>
<tr>
<td>OddsRatioExactCL</td>
<td>Exact confidence limits for the odds ratio</td>
<td>EXACT</td>
<td>OR (2 × 2 table)</td>
</tr>
<tr>
<td>OneWayChiSq</td>
<td>One-way chi-square test</td>
<td>TABLES</td>
<td>CHISQ (one-way table)</td>
</tr>
<tr>
<td>OneWayChiSqMC</td>
<td>Monte Carlo exact test for one-way chi-square</td>
<td>EXACT</td>
<td>CHISQ / MC (one-way table)</td>
</tr>
<tr>
<td>OneWayFreqs</td>
<td>One-way frequencies</td>
<td>PROC or TABLES</td>
<td>(no TABLES stmt) (one-way table)</td>
</tr>
<tr>
<td>OneWayLRChiSq</td>
<td>One-way likelihood ratio chi-square test</td>
<td>TABLES</td>
<td>CHISQ(LRCHI) (one-way table)</td>
</tr>
<tr>
<td>OverallKappa</td>
<td>Overall simple kappa</td>
<td>TABLES</td>
<td>AGREE (h × 2 × 2 table)</td>
</tr>
<tr>
<td>OverallKappas</td>
<td>Overall kappa coefficients</td>
<td>TABLES</td>
<td>AGREE (h × r × r, &gt; 2)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>PdiffCLs</td>
<td>Proportion difference confidence limits</td>
<td>TABLES</td>
<td>RISKDIFF(CL=)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PdiffEquiv</td>
<td>Equivalence analysis for the proportion difference</td>
<td>TABLES</td>
<td>RISKDIFF(EQUIV)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PdiffEquivLimits</td>
<td>Equivalence limits for the proportion difference</td>
<td>TABLES</td>
<td>RISKDIFF(EQUIV)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PdiffEquivTest</td>
<td>Equivalence test for the proportion difference</td>
<td>TABLES</td>
<td>RISKDIFF(EQUIV)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PdiffNoninf</td>
<td>Noninferiority test for the proportion difference</td>
<td>TABLES</td>
<td>RISKDIFF(NONINF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PdiffSup</td>
<td>Superiority test for the proportion difference</td>
<td>TABLES</td>
<td>RISKDIFF(SUP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PdiffTest</td>
<td>Proportion difference test</td>
<td>TABLES</td>
<td>RISKDIFF(EQUAL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>PearsonChiSq</td>
<td>Pearson chi-square exact test</td>
<td>EXACT</td>
<td>PCHI</td>
</tr>
<tr>
<td>PearsonChiSqMC</td>
<td>Monte Carlo exact test for Pearson chi-square</td>
<td>EXACT</td>
<td>PCHI / MC</td>
</tr>
<tr>
<td>PearsonCorr</td>
<td>Pearson correlation</td>
<td>TEST</td>
<td>PCORR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>PCORR</td>
</tr>
<tr>
<td>PearsonCorrMC</td>
<td>Monte Carlo exact test for Pearson correlation</td>
<td>EXACT</td>
<td>PCORR / MC</td>
</tr>
<tr>
<td>PearsonCorrTest</td>
<td>Pearson correlation test</td>
<td>TEST</td>
<td>PCORR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>PCORR</td>
</tr>
<tr>
<td>PlCorr</td>
<td>Polychoric correlation</td>
<td>TEST</td>
<td>PLCORR</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>PlCorrTest</td>
<td>Polychoric correlation test</td>
<td>TEST</td>
<td>PLCORR</td>
</tr>
<tr>
<td>RelativeRisks</td>
<td>Relative risk estimates</td>
<td>TABLES</td>
<td>RELRISK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MEASURES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>RelRisk1ExactCL</td>
<td>Exact confidence limits for column 1</td>
<td>EXACT</td>
<td>RELRISK</td>
</tr>
<tr>
<td></td>
<td>relative risk</td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>RelRisk2ExactCL</td>
<td>Exact confidence limits for column 2</td>
<td>EXACT</td>
<td>RELRISK</td>
</tr>
<tr>
<td></td>
<td>relative risk</td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>RiskDiffCol1</td>
<td>Column 1 risk estimates</td>
<td>TABLES</td>
<td>RISKDIFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>RiskDiffCol2</td>
<td>Column 2 risk estimates</td>
<td>TABLES</td>
<td>RISKDIFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2 × 2 table)</td>
</tr>
<tr>
<td>RowScores</td>
<td>Row scores</td>
<td>TABLES</td>
<td>SCOROOUT</td>
</tr>
<tr>
<td>SomersDCR</td>
<td>Somers' $D(C</td>
<td>R)$</td>
<td>TEST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>SMDCR</td>
</tr>
<tr>
<td>SomersDCRM C</td>
<td>Monte Carlo exact test for Somers'</td>
<td>EXACT</td>
<td>SMDCR / MC</td>
</tr>
<tr>
<td></td>
<td>$D(C</td>
<td>R)$</td>
<td></td>
</tr>
<tr>
<td>SomersDCRTest</td>
<td>Somers' $D(C</td>
<td>R)$ test</td>
<td>TEST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>SMDCR</td>
</tr>
<tr>
<td>SomersDRC</td>
<td>Somers' $D(R</td>
<td>C)$</td>
<td>TEST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>SMDRC</td>
</tr>
<tr>
<td>SomersDRCMC</td>
<td>Monte Carlo exact test for Somers'</td>
<td>EXACT</td>
<td>SMDRC / MC</td>
</tr>
<tr>
<td></td>
<td>$D(R</td>
<td>C)$</td>
<td></td>
</tr>
<tr>
<td>SomersDRCTest</td>
<td>Somers' $D(R</td>
<td>C)$ test</td>
<td>TEST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>SMDRC</td>
</tr>
<tr>
<td>SpearmanCorr</td>
<td>Spearman correlation</td>
<td>TEST</td>
<td>SCORR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or EXACT</td>
<td>SCORR</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>SpearmanCorrMC</td>
<td>Monte Carlo exact test for Spearman correlation</td>
<td>EXACT</td>
<td>SCORR / MC</td>
</tr>
<tr>
<td>SpearmanCorrTest</td>
<td>Spearman correlation test</td>
<td>TEST or EXACT</td>
<td>SCORR</td>
</tr>
<tr>
<td>SymmetryTest</td>
<td>Test of symmetry</td>
<td>TABLES</td>
<td>AGREE</td>
</tr>
<tr>
<td>TauB</td>
<td>Kendall’s tau-</td>
<td>TEST or EXACT</td>
<td>KENTB</td>
</tr>
<tr>
<td>TauBMC</td>
<td>Monte Carlo exact test for Kendall’s tau-</td>
<td>EXACT</td>
<td>KENTB / MC</td>
</tr>
<tr>
<td>TauBTest</td>
<td>Kendall’s tau- test</td>
<td>TEST or EXACT</td>
<td>KENTB</td>
</tr>
<tr>
<td>TauC</td>
<td>Stuart’s tau-</td>
<td>TEST or EXACT</td>
<td>STUTC</td>
</tr>
<tr>
<td>TauCMC</td>
<td>Monte Carlo exact test for Stuart’s tau-</td>
<td>EXACT</td>
<td>STUTC / MC</td>
</tr>
<tr>
<td>TauCTest</td>
<td>Stuart’s tau- test</td>
<td>TEST or EXACT</td>
<td>STUTC</td>
</tr>
<tr>
<td>TrendTest</td>
<td>Cochran-Armitage trend test</td>
<td>TABLES</td>
<td>TREND</td>
</tr>
<tr>
<td>TrendTestMC</td>
<td>Monte Carlo exact test for trend</td>
<td>EXACT</td>
<td>TREND / MC</td>
</tr>
<tr>
<td>WtKappa</td>
<td>Weighted kappa coefficient</td>
<td>TEST or EXACT</td>
<td>WTKAP</td>
</tr>
<tr>
<td>WtKappaMC</td>
<td>Monte Carlo exact test for weighted kappa coefficient</td>
<td>EXACT</td>
<td>WTKAP / MC</td>
</tr>
<tr>
<td>WtKappaTest</td>
<td>Weighted kappa test</td>
<td>TEST or EXACT</td>
<td>WTKAP</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>Summary of descriptive statistics for variables across all observations and within groups of observations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table A13.10  ODS Table Names Produced by the PLOT Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot</td>
<td>Single plot graph</td>
<td></td>
</tr>
<tr>
<td>Overlaid</td>
<td>Two or more plots on a single set of axes</td>
<td>OVERLAY</td>
</tr>
</tbody>
</table>

### Table A13.11  ODS Table Names Produced by the REPORT Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Detail report, summary report, or combination of both detail and summary information report</td>
</tr>
</tbody>
</table>

### Table A13.12  ODS Table Names Produced by the SQL Procedure

**For detailed information, see the SAS SQL Procedure User’s Guide.**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_Results</td>
<td>SAS data file or a SAS data view</td>
</tr>
</tbody>
</table>
### Table A13.13  ODS Table Names Produced by the TABULATE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Descriptive statistics in tabular format that use some or all of the variables in a data set</td>
</tr>
</tbody>
</table>

### Table A13.14  ODS Table Names Produced by the TIMEPLOT Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot</td>
<td>Single plot graph</td>
<td>Omit the OVERLAY option</td>
</tr>
<tr>
<td>OverlaidPlot</td>
<td>Two or more plots on a single set of axes</td>
<td>OVERLAY</td>
</tr>
</tbody>
</table>

### Table A13.15  ODS Table Names Produced by the UNIVARIATE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasicIntervals</td>
<td>Confidence intervals for mean, standard deviation, variance</td>
<td>CIBASIC</td>
</tr>
<tr>
<td>BasicMeasures</td>
<td>Measures of location and variability</td>
<td>Default</td>
</tr>
<tr>
<td>ExtremeObs</td>
<td>Extreme observations</td>
<td>Default</td>
</tr>
<tr>
<td>ExtremeValues</td>
<td>Extreme values</td>
<td>NEXTRAVAL=</td>
</tr>
<tr>
<td>Frequencies</td>
<td>Frequencies</td>
<td>FREQ</td>
</tr>
<tr>
<td>LocationCounts</td>
<td>Counts used for sign test and signed rank test</td>
<td>LOCCOUNT</td>
</tr>
<tr>
<td>Missing Values</td>
<td>Missing values</td>
<td>Default, if missing values exist</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Modes</td>
<td>Modes</td>
<td>MODES</td>
</tr>
<tr>
<td>Moments</td>
<td>Sample moments</td>
<td>Default</td>
</tr>
<tr>
<td>Plots</td>
<td>Line printer plots</td>
<td>PLOTS</td>
</tr>
<tr>
<td>Quantiles</td>
<td>Quantiles</td>
<td>Default</td>
</tr>
<tr>
<td>RobustScale</td>
<td>Robust measures of scale</td>
<td>ROBUSTSCALE</td>
</tr>
<tr>
<td>SSPlots</td>
<td>Line printer side-by-side box plot</td>
<td>PLOTS with BY statement</td>
</tr>
<tr>
<td>TestsForLocation</td>
<td>Tests for location</td>
<td>Default</td>
</tr>
<tr>
<td>TestsForNormality</td>
<td>Tests for normality</td>
<td>NORMALTEST</td>
</tr>
<tr>
<td>TrimmedMeans</td>
<td>Trimmed means</td>
<td>TRIMMED=</td>
</tr>
<tr>
<td>WinsorizedMeans</td>
<td>Winsorized means</td>
<td>WINSORIZED=</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the HISTOGRAM Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bins</td>
<td>Histogram bins</td>
<td>MIDPERCENTS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>secondary option</td>
</tr>
<tr>
<td>FitQuantiles</td>
<td>Quantiles of fitted distribution</td>
<td>Any distribution option</td>
</tr>
<tr>
<td>GoodnessOfFit</td>
<td>Goodness-of-fit tests for fitted distribution</td>
<td>Any distribution option</td>
</tr>
<tr>
<td>HistogramBins</td>
<td>Histogram bins</td>
<td>MIDPERCENTS option</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates for fitted distribution</td>
<td>Any distribution option</td>
</tr>
</tbody>
</table>
**ODS Table Names Produced by Base SAS High-Performance Procedures**

*Table A13.16  ODS Table Names Produced by Base SAS High-Performance Procedures*

For detailed information, see the procedures in *Base SAS Procedures Guide: High-Performance Procedures*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPBIN</td>
<td>ODS Tables Produced by PROC HPBIN</td>
</tr>
<tr>
<td>HPCORR</td>
<td>ODS Tables Produced by PROC HPCORR</td>
</tr>
<tr>
<td>HPIMPUTE</td>
<td>ODS Tables Produced by PROC HPIMPUTE</td>
</tr>
<tr>
<td>HPSAMPLE</td>
<td>ODS Tables Produced by PROC HPSAMPLE</td>
</tr>
</tbody>
</table>

**ODS Table Names Produced by Base SAS Statistical Procedures**

*Table A13.17  ODS Table Names Produced by Base SAS Statistical Procedures*

For detailed information, see the procedures in *Base SAS Procedures Guide: Statistical Procedures*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORR</td>
<td>ODS Tables Produced by PROC CORR</td>
</tr>
<tr>
<td>FREQ</td>
<td>ODS Tables Produced by PROC FREQ</td>
</tr>
<tr>
<td>UNIVARIATE</td>
<td>ODS Tables Produced by PROC UNIVARIATE</td>
</tr>
</tbody>
</table>
### ODS Table Names Produced by SAS/STAT Procedures

For detailed information, see the procedures in *SAS/STAT User’s Guide*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACECLUS</td>
<td>ODS Tables Produced by PROC ACECLUS</td>
</tr>
<tr>
<td>ADAPTIVEREG</td>
<td>ODS Tables Produced by PROC ADAPTIVEREG</td>
</tr>
<tr>
<td>ANOVA</td>
<td>ODS Tables Produced by PROC ANOVA</td>
</tr>
<tr>
<td>BCHOICE</td>
<td>ODS Tables Produced by PROC BCHOICE</td>
</tr>
<tr>
<td>CALIS</td>
<td>ODS Tables Produced by PROC CALIS</td>
</tr>
<tr>
<td>CANCORR</td>
<td>ODS Tables Produced by PROC CANCORR</td>
</tr>
<tr>
<td>CANDISC</td>
<td>ODS Tables Produced by PROC CANDISC</td>
</tr>
<tr>
<td>CATMOD</td>
<td>ODS Tables Produced by PROC CATMOD</td>
</tr>
<tr>
<td>CAUSALGRAPH</td>
<td>ODS Tables Produced by PROC CAUSALGRAPH</td>
</tr>
<tr>
<td>CAUSALMED</td>
<td>ODS Tables Produced by PROC CAUSALMED</td>
</tr>
<tr>
<td>CAUSALTRT</td>
<td>ODS Tables Produced by PROC CAUSALTRT</td>
</tr>
<tr>
<td>CLUSTER</td>
<td>ODS Tables Produced by PROC CLUSTER</td>
</tr>
<tr>
<td>CORRESP</td>
<td>ODS Tables Produced by PROC CORRESP</td>
</tr>
<tr>
<td>DISCRIM</td>
<td>ODS Tables Produced by PROC DISCRIM</td>
</tr>
<tr>
<td>FACTOR</td>
<td>ODS Tables Produced by PROC FACTOR</td>
</tr>
<tr>
<td>FASTCLUS</td>
<td>ODS Tables Produced by PROC FASTCLUS</td>
</tr>
<tr>
<td>FMM</td>
<td>ODS Tables Produced by PROC FMM</td>
</tr>
<tr>
<td>FREQ</td>
<td>ODS Tables Produced by PROC FREQ</td>
</tr>
</tbody>
</table>
For detailed information, see the procedures in *SAS/STAT User’s Guide*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAM</td>
<td>ODS Tables Produced by PROC GAM</td>
</tr>
<tr>
<td>GAMPL</td>
<td>ODS Tables Produced by PROC GAMPL</td>
</tr>
<tr>
<td>GEE</td>
<td>ODS Tables Produced by PROC GEE</td>
</tr>
<tr>
<td>GENMOD</td>
<td>ODS Table Produced by PROC GENMOD</td>
</tr>
<tr>
<td>GLIMMIX</td>
<td>ODS Tables Produced by Proc GLIMMIX</td>
</tr>
<tr>
<td>GLM</td>
<td>ODS Tables Produced by PROC GLM</td>
</tr>
<tr>
<td>GLMMOD</td>
<td>ODS Tables Produced by PROC GLMMOD</td>
</tr>
<tr>
<td>GLMPOWER</td>
<td>ODS Tables Produced by PROC GLMPOWER</td>
</tr>
<tr>
<td>GLMSELECT</td>
<td>ODS Tables Produced by PROC GLMSELECT</td>
</tr>
<tr>
<td>HPCANDISC</td>
<td>ODS Tables Produced by PROC HPCANDISC</td>
</tr>
<tr>
<td>HPFMM</td>
<td>ODS Tables Produced by PROC HPFMM</td>
</tr>
<tr>
<td>HPGENSELECT</td>
<td>ODS Tables Produced by PROC HPGENSELECT</td>
</tr>
<tr>
<td>HPLMIXED</td>
<td>ODS Tables Produced by PROC HPLMIXED</td>
</tr>
<tr>
<td>HPLOGISTIC</td>
<td>ODS Tables Produced by PROC HPLOGISTIC</td>
</tr>
<tr>
<td>HPMIXED</td>
<td>ODS Tables Produced by PROC HPMIXED</td>
</tr>
<tr>
<td>HPNLMOD</td>
<td>ODS Tables Produced by PROC HPNLMOD</td>
</tr>
<tr>
<td>HPPLS</td>
<td>ODS Tables Produced by PROC HPPLS</td>
</tr>
<tr>
<td>HPPRINCOMP</td>
<td>ODS Tables Produced by PROC HPPRINCOMP</td>
</tr>
<tr>
<td>HPQUANTSELECT</td>
<td>ODS Tables Produced by PROC HPQUANTSELECT</td>
</tr>
<tr>
<td>HPREG</td>
<td>ODS Tables Produced by PROC HPREG</td>
</tr>
<tr>
<td>HPSPLIT</td>
<td>ODS Tables Produced by PROC HPSPLIT</td>
</tr>
<tr>
<td>ICLIFETEST</td>
<td>ODS Tables Produced by PROC ICLIFETEST</td>
</tr>
<tr>
<td>ICPHREG</td>
<td>ODS Tables Produced by PROC ICPHREG</td>
</tr>
</tbody>
</table>
For detailed information, see the procedures in *SAS/STAT User’s Guide*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names Produced by PROC</th>
<th>ODS Table Names Produced by PROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>INBREED</td>
<td>ODS Tables Produced by PROC INBREED</td>
<td></td>
</tr>
<tr>
<td>IRT</td>
<td>ODS Tables Produced by PROC IRT</td>
<td></td>
</tr>
<tr>
<td>KDE</td>
<td>ODS Tables Produced by PROC KDE</td>
<td></td>
</tr>
<tr>
<td>KRIGE2D</td>
<td>ODS Tables Produced by PROC KRIGE2d</td>
<td></td>
</tr>
<tr>
<td>LATTICE</td>
<td>ODS Tables Produced by PROC LATTICE</td>
<td></td>
</tr>
<tr>
<td>LIFEREG</td>
<td>ODS Tables Produced by PROC LIFEREG</td>
<td></td>
</tr>
<tr>
<td>LIFETEST</td>
<td>ODS Test Produced by PROC LIFETEST</td>
<td></td>
</tr>
<tr>
<td>LOESS</td>
<td>ODS Tables Produced by PROC LOESS</td>
<td></td>
</tr>
<tr>
<td>LOGISTIC</td>
<td>ODS Tables Produced by PROC LOGISTIC</td>
<td></td>
</tr>
<tr>
<td>MCMC</td>
<td>ODS Tables Produced by PROC MCMC</td>
<td></td>
</tr>
<tr>
<td>MDS</td>
<td>ODS Tables Produced by PROC MDS</td>
<td></td>
</tr>
<tr>
<td>MI</td>
<td>ODS Table Produced by PROC MI</td>
<td></td>
</tr>
<tr>
<td>MIANALYZE</td>
<td>ODS Tables Produced by PROC MIANALYZE</td>
<td></td>
</tr>
<tr>
<td>MIXED</td>
<td>ODS Tables Produced by PROC MIXED</td>
<td></td>
</tr>
<tr>
<td>MODECLUS</td>
<td>ODS Tables Produced by PROC MODECLUS</td>
<td></td>
</tr>
<tr>
<td>MULTTEST</td>
<td>ODS Tables Produced by PROC MULTTEST</td>
<td></td>
</tr>
<tr>
<td>NESTED</td>
<td>ODS Tables Produced by PROC NESTED</td>
<td></td>
</tr>
<tr>
<td>NLIN</td>
<td>ODS Tables Produced by PROC NLIN</td>
<td></td>
</tr>
<tr>
<td>NLMIXED</td>
<td>ODS Tables Produced by PROC NLMIXED</td>
<td></td>
</tr>
<tr>
<td>NPAR1WAY</td>
<td>ODS Tables Produced by PROC NPAR1WAY</td>
<td></td>
</tr>
<tr>
<td>ORTHOREG</td>
<td>ODS Tables Produced by PROC ORTHOREG</td>
<td></td>
</tr>
<tr>
<td>PHREG</td>
<td>ODS Tables Produced by PROC PHREG</td>
<td></td>
</tr>
<tr>
<td>PLAN</td>
<td>ODS Tables Produced by PROC PLAN</td>
<td></td>
</tr>
<tr>
<td>Procedure</td>
<td>ODS Table Names</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>PLM</td>
<td>ODS Tables Produced by PROC PLM</td>
<td></td>
</tr>
<tr>
<td>PLS</td>
<td>ODS Tables Produced by PROC PLS</td>
<td></td>
</tr>
<tr>
<td>POWER</td>
<td>ODS Tables Produced by PROC POWER</td>
<td></td>
</tr>
<tr>
<td>PRINCOMP</td>
<td>ODS Tables Produced by PROC PRINCOMP</td>
<td></td>
</tr>
<tr>
<td>PRINQUAL</td>
<td>ODS Tables Produced by PROC PRINQUAL</td>
<td></td>
</tr>
<tr>
<td>PROBIT</td>
<td>ODS Tables Produced by PROC PROBIT</td>
<td></td>
</tr>
<tr>
<td>PSMATCH</td>
<td>ODS Tables Produced by PROC PSMATCH</td>
<td></td>
</tr>
<tr>
<td>QUANTLIFE</td>
<td>ODS Tables Produced by PROC QUANTLIFE</td>
<td></td>
</tr>
<tr>
<td>QUANTREG</td>
<td>ODS Tables Produced by PROC QUANTREG</td>
<td></td>
</tr>
<tr>
<td>QUANTSELECT</td>
<td>ODS Tables Produced by PROC QUANTSELECT</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>ODS Tables Produced by PROC REG</td>
<td></td>
</tr>
<tr>
<td>RMSTREG</td>
<td>ODS Tables Produced by PROC RMSTREG</td>
<td></td>
</tr>
<tr>
<td>ROBUSTREG</td>
<td>ODS Tables Produced by PROC ROBUSTREG</td>
<td></td>
</tr>
<tr>
<td>RSREG</td>
<td>ODS Tables Produced by PROC RSREG</td>
<td></td>
</tr>
<tr>
<td>SEQDESIGN</td>
<td>ODS Tables Produced by PROC SEQDESIGN</td>
<td></td>
</tr>
<tr>
<td>SEQTEST</td>
<td>ODS Tables Produced by PROC SEQTEST</td>
<td></td>
</tr>
<tr>
<td>SIM2D</td>
<td>ODS Tables Produced by PROC SIM2D</td>
<td></td>
</tr>
<tr>
<td>SPP</td>
<td>ODS Tables Produced by PROC SPP</td>
<td></td>
</tr>
<tr>
<td>STDIZE</td>
<td>ODS Tables Produced by PROC STDIZE</td>
<td></td>
</tr>
<tr>
<td>STDRATE</td>
<td>ODS Tables Produced by PROC STDRATE</td>
<td></td>
</tr>
<tr>
<td>STEPDISC</td>
<td>ODS Tables Produced by PROC STEPDISC</td>
<td></td>
</tr>
<tr>
<td>SURVEYFREQ</td>
<td>ODS Tables Produced by PROC SURVEYFREQ</td>
<td></td>
</tr>
<tr>
<td>SURVEYIMPUTE</td>
<td>ODS Tables Produced by PROC SURVEYIMPUTE</td>
<td></td>
</tr>
</tbody>
</table>

For detailed information, see the procedures in *SAS/STAT User’s Guide*
For detailed information, see the procedures in *SAS/STAT User’s Guide*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>SURVEYLOGISTIC</td>
<td>ODS Tables Produced by PROC SURVEYLOGISTIC</td>
</tr>
<tr>
<td>SURVEYMEANS</td>
<td>ODS Tables Produced by PROC SURVEYMEANS</td>
</tr>
<tr>
<td>SURVEYPHREG</td>
<td>ODS Tables Produced by PROC SURVEYPHREG</td>
</tr>
<tr>
<td>SURVEYREG</td>
<td>ODS Tables Produced by PROC SURVEYREG</td>
</tr>
<tr>
<td>SURVEYSELECT</td>
<td>ODS Tables Produced by PROC SURVEYSELECT</td>
</tr>
<tr>
<td>TPSLINE</td>
<td>ODS Tables Produced by PROC TPSLINE</td>
</tr>
<tr>
<td>TRANSREG</td>
<td>ODS Tables Produced by PROC TRANSREG</td>
</tr>
<tr>
<td>TREE</td>
<td>ODS Tables Produced by PROC TREE</td>
</tr>
<tr>
<td>TTEST</td>
<td>ODS Tables Produced by PROC TTEST</td>
</tr>
<tr>
<td>VARCLUS</td>
<td>ODS Tables Produced by PROC VARCLUS</td>
</tr>
<tr>
<td>VARCOMP</td>
<td>ODS Tables Produced by PROC VARCOMP</td>
</tr>
<tr>
<td>VARIOGRAM</td>
<td>ODS Tables Produced by PROC VARIOGRAM</td>
</tr>
</tbody>
</table>

### ODS Table Names Produced by SAS/STAT High-Performance Procedures

*Table A13.19 ODS Table Names Produced by SAS/STAT High-Performance Procedures*

For detailed information, see the procedures in *SAS/STAT User’s Guide: High-Performance Procedures*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMPL</td>
<td>ODS Tables Produced by PROC GAMPL</td>
</tr>
<tr>
<td>HPCANDISC</td>
<td>ODS Tables Produced by PROC HPCANDISC</td>
</tr>
<tr>
<td>HPFMM</td>
<td>ODS Tables Produced by PROC HPFMM</td>
</tr>
</tbody>
</table>
For detailed information, see the procedures in **SAS/STAT User’s Guide: High-Performance Procedures**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPGENSELECT</td>
<td>ODS Tables Produced by PROC HPGENSELECT</td>
</tr>
<tr>
<td>HPLMIXED</td>
<td>ODS Tables Produced by PROC HPLMIXED</td>
</tr>
<tr>
<td>HPLOGISTIC</td>
<td>ODS Tables Produced by PROC HPLOGISTIC</td>
</tr>
<tr>
<td>HPNLMOD</td>
<td>ODS Tables Produced by PROC HPNLMOD</td>
</tr>
<tr>
<td>HPPLS</td>
<td>ODS Tables Produced by PROC HPPLS</td>
</tr>
<tr>
<td>HPPRINCOMP</td>
<td>ODS Tables Produced by PROC HPPRINCOMP</td>
</tr>
<tr>
<td>HPQUANTSELECT</td>
<td>ODS Tables Produced by PROC HPQUANTSELECT</td>
</tr>
<tr>
<td>HPREG</td>
<td>ODS Tables Produced by PROC HPREG</td>
</tr>
<tr>
<td>HPSPLIT</td>
<td>ODS Tables Produced by PROC HPSPLIT</td>
</tr>
</tbody>
</table>

**Table A13.20**  ODS Table Names Produced by SAS Enterprise Miner High-Performance Procedures

For detailed information, see the procedures in **SAS Enterprise Miner: High-Performance Procedures**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP4SCORE</td>
<td>ODS Tables Produced by PROC HP4SCORE</td>
</tr>
<tr>
<td>HPCLUS</td>
<td>ODS Tables Produced by PROC HPCLUS</td>
</tr>
<tr>
<td>HPDECIDE</td>
<td>ODS Tables Produced by PROC HPDECIDE</td>
</tr>
<tr>
<td>HPFOREST</td>
<td>ODS Tables Produced by PROC HPFOREST</td>
</tr>
<tr>
<td>HPNEURAL</td>
<td>ODS Tables Produced by PROC HPNEURAL</td>
</tr>
</tbody>
</table>
### ODS Table Names Produced by SAS/ETS Procedures

This table lists the output object table names that 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>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names Produced by SAS/ETS Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA</td>
<td>ODS Tables Produced by PROC ARIMA</td>
</tr>
<tr>
<td>AUTOREG</td>
<td>ODS Tables Produced by PROC AUTOREG</td>
</tr>
<tr>
<td>COPULA</td>
<td>ODS Tables Produced by PROC COPULA</td>
</tr>
<tr>
<td>COUNTREG</td>
<td>ODS Tables Produced by PROC COUNTREG</td>
</tr>
<tr>
<td>ENTROPY</td>
<td>ODS Tables Produced by PROC ENTROPY</td>
</tr>
<tr>
<td>ESM</td>
<td>ODS Tables Produced by PROC ESM</td>
</tr>
<tr>
<td>HPCOUNTREG</td>
<td>ODS Tables Produced by PROC HPCOUNTREG</td>
</tr>
<tr>
<td>HPPANEL</td>
<td>ODS Tables Produced by PROC HPPANEL</td>
</tr>
<tr>
<td>HPQLIM</td>
<td>ODS Tables Produced by PROC HPQLIM</td>
</tr>
<tr>
<td>LOAN</td>
<td>ODS Tables Produced by PROC LOAN</td>
</tr>
</tbody>
</table>
For detailed information, see the procedures in *SAS/ETS User’s Guide*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDC</td>
<td>ODS Tables Produced by PROC MDC</td>
</tr>
<tr>
<td>PANEL</td>
<td>ODS Tables Produced by PROC PANEL</td>
</tr>
<tr>
<td>QLIM</td>
<td>ODS Tables Produced by PROC QLIM</td>
</tr>
<tr>
<td>SIMILARITY</td>
<td>ODS Tables Produced by PROC SIMILARITY</td>
</tr>
<tr>
<td>SIMLIN</td>
<td>ODS Tables Produced by PROC SIMLIN</td>
</tr>
<tr>
<td>SPATIALREG</td>
<td>ODS Tables Produced by PROC SPATIALREG</td>
</tr>
<tr>
<td>SPECTRA</td>
<td>ODS Tables Produced by PROC SPECTRA</td>
</tr>
<tr>
<td>SSM</td>
<td>ODS Tables Produced by PROC SSM</td>
</tr>
<tr>
<td>STATESPACE</td>
<td>ODS Tables Produced by PROC STATESPACE</td>
</tr>
<tr>
<td>SYSLIN</td>
<td>ODS Tables Produced by PROC SYSLIN</td>
</tr>
<tr>
<td>TIMEDATA</td>
<td>ODS Tables Produced by PROC TIMEDATA</td>
</tr>
<tr>
<td>TIMESERIES</td>
<td>ODS Tables Produced by PROC TIMESERIES</td>
</tr>
<tr>
<td>TSCSREG</td>
<td>ODS Tables Produced by PROC TSCSREG</td>
</tr>
<tr>
<td>UCM</td>
<td>ODS Tables Produced by PROC UCM</td>
</tr>
<tr>
<td>VARMAX</td>
<td>ODS Tables Produced by PROC VARMAX</td>
</tr>
<tr>
<td>X11</td>
<td>ODS Tables Produced by PROC X11</td>
</tr>
<tr>
<td>X13</td>
<td>ODS Tables Produced by PROC X13</td>
</tr>
</tbody>
</table>
### ODS Table Names Produced by SAS/ETS High-Performance Procedures

**Table A13.22** ODS Table Names Produced by SAS/ETS High-Performance Procedures

For detailed information, see the procedures in *SAS/ETS User’s Guide: High-Performance Procedures*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPCOUNTREG</td>
<td>ODS Tables Produced by PROC HPCOUNTREG</td>
</tr>
<tr>
<td>HPPANEL</td>
<td>ODS Tables Produced by PROC HPPANEL</td>
</tr>
<tr>
<td>HPQLIM</td>
<td>ODS Tables Produced by PROC HPQLIM</td>
</tr>
</tbody>
</table>

### ODS Table Names Produced by SAS/QC Procedures

**Table A13.23** ODS Table Names Produced by SAS/QC Procedures

For detailed information, see the procedures in *SAS/QC User’s Guide*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACTEX</td>
<td>ODS Tables Produced by PROC FACTEX</td>
</tr>
<tr>
<td>MVPMODEL</td>
<td>ODS Tables Produced by PROC MVPMODEL</td>
</tr>
<tr>
<td>OPTEX</td>
<td>ODS Tables Produced by PROC OPTEX</td>
</tr>
<tr>
<td>RAREEVENTS</td>
<td>ODS Tables Produced by PROC RAREEVENTS</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>ODS Tables Produced by PROC RELIABILITY</td>
</tr>
</tbody>
</table>
# ODS Table Names Produced by SAS/OR Procedures

For detailed information, see the procedures in the following documents:
- SAS/OR User’s Guide: Mathematical Programming
- SAS/OR User’s Guide: Local Search Optimization

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTREE</td>
<td>ODS Tables Produced by PROC DTREE</td>
</tr>
<tr>
<td>OPTLP</td>
<td>ODS Tables Produced by PROC OPTLP</td>
</tr>
<tr>
<td>OPTLSO</td>
<td>ODS Tables Produced by PROC OPTLSO</td>
</tr>
<tr>
<td>OPTMILP</td>
<td>ODS Tables Produced by PROC OPTMILP</td>
</tr>
<tr>
<td>OPTMODEL</td>
<td>ODS Tables Produced by PROC OPTMODEL</td>
</tr>
<tr>
<td>OPTNET</td>
<td>ODS Tables Produced by PROC OPTNET</td>
</tr>
<tr>
<td>OPTQP</td>
<td>ODS Tables Produced by PROC OPTQP</td>
</tr>
</tbody>
</table>
### ODS Table Names Produced by Graph Algorithms and Network Analysis Procedure

**Table A13.25**  
*ODS Table Names Produced by Graph Algorithms and Network Analysis Procedure*

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ODS Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTGRAPH</td>
<td>ODS Tables Produced by PROC</td>
</tr>
<tr>
<td></td>
<td>OPTGRAPH</td>
</tr>
</tbody>
</table>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.