SAS® 9.1 Output Delivery System
User’s Guide
# Contents

- What's New vii
- Overview vii
- Details vii

## PART 1  
**Introduction** 1

- Chapter 1 △ Getting Started with the Output Delivery System 3
  - Welcome to the Output Delivery System 3
  - A Quick Start to Using ODS 3
  - Where to Go from Here 9

## PART 2  
**Concepts** 11

- Chapter 2 △ Output Delivery System: Basic Concepts 13
  - What Is the Output Delivery System? 13
  - Gallery of ODS Samples 14
  - Commonly Used ODS Terminology 21
  - How Does ODS Work? 22
  - What Are the ODS Destinations? 25
  - What Are Table Definitions, Table Elements, and Table Attributes? 29
  - What Are Style Definitions, Style Elements, and Style Attributes? 29
  - Changing SAS Registry Settings for ODS 31
  - Customized ODS Output 34
  - Summary of ODS 37

- Chapter 3 △ Output Delivery System and the DATA Step 39
  - Why Use ODS with the DATA Step? 39
  - How ODS Works with the DATA Step 40
  - Syntax for ODS Enhanced Features in a DATA Step 41
  - Examples 41

## PART 3  
**ODS Language Statements** 59

- Chapter 4 △ Introduction to ODS Language Statements 61
  - Definition of ODS Statements 61
  - Types of ODS Statements 61
  - ODS Statement Category Descriptions 62
  - ODS Statements by Category 63

- Chapter 5 △ Dictionary of ODS Language Statements 67

## PART 4  
The DOCUMENT Procedure 209
What’s New

Overview

The Output Delivery System (ODS) provides an almost limitless number of choices for reporting and displaying analytical results with a greater variety of formatting selections and output destinations.

SAS 9 and 9.1 provide an array of markup languages including HTML4 and XML. The TEMPLATE procedure and the new tagset template enable you to modify any markup language that SAS provides, or to create your own markup language for output.

The new experimental ODS GRAPHICS statement enables you to produce graphics output.

The new DOCUMENT procedure enables you to customize or modify your output hierarchy and replay your output to different destinations without rerunning the PROC or DATA step.

Note:

□ This section describes the features of the SAS Output Delivery System that are new or enhanced since SAS 8.2.

□ z/OS is the successor to the OS/390 operating system. SAS 9.1 is supported on both OS/390 and z/OS operating systems and, throughout this document, any reference to z/OS also applies to OS/390, unless otherwise stated.

Details

SASEDOC Engine

The new SASEDOC libname engine enables you to associate a SAS libref (library reference) with one or more output objects that are stored in an ODS document as a SAS data set.
ODS Statements

The following ODS statements are new:

ODS CHTML
    produces compact, minimal HTML output with no style information.

ODS CSVALL
    produces output that contains columns of data values that are separated by commas. ODS CSVALL produces tabular output with titles, notes, and bylines.

ODS DECIMAL_ALIGN “ODS DECIMAL_ALIGN Statement” on page 85
    aligns values by the decimal point in numeric columns when no justification is specified.

ODS DOCBOOK
    produces XML output that conforms to the DocBook DTD by OASIS.

ODS DOCUMENT
    produces a hierarchy of output objects that enables you to create multiple ODS output formats without rerunning a PROC or DATA step.

ODS GRAPHICS “ODS GRAPHICS Statement (Experimental)” on page 92
    (Experimental)
    enables ODS automatic graphic capabilities.

ODS HTMLCSS
    produces HTML output with cascading stylesheets that is similar to ODS HTML output.

ODS IMODE
    produces HTML output as a column of output that is separated by lines.

ODS MARKUP
    produces SAS output that is formatted using one of many different markup languages.

ODS PCL
    produces printable output for PCL (HP LaserJet) files.

ODS PDF
    produces PDF output.

ODS PHTML
    produces basic HTML output that uses twelve style elements and no class attributes.

ODS PS
    produces PostScript (PS) output.

ODS USEGOPT “ODS USEGOPT Statement” on page 202
    enables the use of graphics option settings for graphic output.

ODS WML
    produces a Wireless Markup Language (WML) DTD with a simple list of URLs for a table of contents.

□ The ODS PRINTER statement now supports the following options:
    BACKGROUND= on page 161
        specifies whether background colors are printed in text.

    BOOKMARKLIST= on page 162
specifies whether to generate and display the list of bookmarks for a PDF file. 

BOOKMARKGEN= on page 162  
controls the generation of bookmarks in a PDF file.

COLUMN$\mathrm{S=} 
specifies the number of columns to create on each page of output.

TEXT= 
inserts text into your output.

- The ODS RTF statement now supports the following options:
  - COLUMN$\mathrm{S=} 
    specifies the number of columns to create on each page of output.
  - TEXT= 
    inserts text into your output.

- The ODS MARKUP statement now supports the following TAGSETS:
  - GRAPH 
    produces markup for graphical output produced by SAS/GRAPH.
  - SASFMT 
    produces user-defined format markup tags for the XML engine.
  - SASXMISS 
    produces alternate missing-value markup tags for the XML engine.
  - SASXMNSP 
    produces alternate “no space in text” value markup for the XML engine.
  - STATGRAPH 
    produces markup for statistical graphs that are generated by SAS procedures.

**ODS Procedures**

- DOCUMENT procedure
  - The new DOCUMENT procedure enables you to do the following:
    - produce multiple reports with a single run of a procedure or data query.
    - modify and customize your output file hierarchy by rearranging, duplicating, or removing specific tables.
    - modify and customize your output file hierarchy by rearranging, duplicating, or removing the entire output of procedures and data queries.
    - generate output for one or more ODS destinations, using the newly transformed output hierarchy.
    - store the ODS output objects in raw form. The output is kept in the original internal representation as a data component plus a table definition.
  - The new LIB= option in the DOC statement enables you to list documents that are in the specified library.
  - The &#BYLINE, #BYVAL, and #BYVAR directives can now be used in seven of the PROC DOCUMENT statements.
  - The new AFTER option in the OBPAGE statement adds or deletes page breaks after output objects.
- **TEMPLATE procedure**
  - The following new statements are provided in the TEMPLATE procedure.
    - The DEFINE TAGSET statement creates and modifies tagset definitions using a new definition statement and the new tagset attribute statements.
    - The DEFINE EVENT statement determines what is written to the output file using the new definition statement and the new event attribute statements.
  - The new boolean ABSTRACT= attribute can be applied to styles. If this attribute is set to TRUE, then the style will not appear in the CSS files or LaTeX style files.
  - The new ALT, LONGDESC, ACRONYM, and ABBR options in the DEFINE TABLE, DEFINE COLUMN, and DEFINE HEADER statements provide accessibility features in PROC TEMPLATE.
  - The following accessibility attributes have been added to the TEMPLATE procedure.
    - `ABBR=` is a column, header, and footer attribute that provides an abbreviated form of the cell’s content, which can be displayed in place of a cell’s content.
    - `SUMMARY=` is a table attribute that provides a summary of the purpose and structure of a table.
  - The new INDENT style element enables you to specify the distance from the left side of a cell for indentation purposes. The INDENT style element specifies where to begin the text.
  - The `_LABEL_` keyword is now treated as a dynamic variable, and it can be used just like any other dynamic variable in PROC TEMPLATE. Previously, `_LABEL_` was a keyword that could only be used by itself in table and column headers.
  - In the TEMPLATE procedure, if you create HTML4 output, then the BORDERCOLORDARK and BORDERCOLORLIGHT style attributes are ignored because they are not part of the HTML4 standard. If you want a color border, then use the BORDER= style attribute.
PART 1

Introduction

Chapter 1. Getting Started with the Output Delivery System 3
Welcome to the Output Delivery System

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

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

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

A Quick Start to Using ODS

The Purpose of These Examples

The following examples are designed to help you get up and running quickly with ODS. Use them to learn how to produce output that contains more interesting formatting. Then, to learn more about the depth, breadth, and true power of ODS, see “What Is the Output Delivery System?” on page 13.
Creating Listing Output

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

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

data employee_data;
  input IdNumber $ 1-4 LastName $ 9-19 FirstName $ 20-29
          City $ 30-42 State $ 43-44 / Gender $ 1 JobCode $ 9-11 Salary 20-29 @30
          Birth date9. @43 Hired date9. HomePhone $ 54-65;
  format birth hired date9.;
  datalines;
1919 Adams Gerald Stamford CT
  M TA2 34376 15SEP48 07JUN75 203/781-1255
1653 Alexander Susan Bridgeport CT
  F ME2 35108 18OCT52 12AUG78 203/675-7715
1400 Apple Troy New York NY
  ME1 29769 08NOV55 19OCT78 212/586-0808
1350 Arthur Barbara New York NY
  F FA3 32886 03SEP53 01AUG78 718/383-1549
1401 Avery Jerry Paterson NJ
  M TA3 38822 16DEC38 20NOV73 201/732-8787
1499 Barefoot Joseph Princeton NJ
  M ME3 43025 29APR42 10JUN68 201/812-5665
1101 Baucom Walter New York NY
  SCP 18723 09JUN50 04OCT78 212/586-8060
1333 Blair Justin Stamford CT
  M PT2 88606 02APR49 13FEB69 203/781-1777
1402 Blalock Ralph New York NY
  M TA2 32615 20JAN51 05DEC78 718/384-2849
1479 Bostic Marie New York NY
  F TA3 38785 25DEC56 08OCT77 718/384-8816
1403 Bowden Earl Bridgeport CT
  M ME1 28072 31JAN57 24DEC79 203/675-3434
1739 Boyce Jonathan New York NY
  M PT1 66517 28DEC52 30JAN79 212/587-1247
1658 Bradley Jeremy New York NY
  M SCP 17943 11APR55 03MAR80 212/587-3622
1428 Brady Christine Stamford CT
  F PT1 68767 07APR58 19NOV79 203/781-1212
1407 Grant Daniel Mt. Vernon NY
  M PT1 68096 26MAR57 21MAR78 914/468-1616
1114 Green Janice New York NY
  F TA2 32928 21SEP57 30JUN75 212/588-1092
;
```

```
proc print data=employee_data(obs=12);
  id idnumber;
  title 'Personnel Data';
run;
```
Output 1.1 Listing Output

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

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

```sas
ods listing close;
```

Creating Output in HTML Format

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

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

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

```sas
ods listing close;
```

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

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

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

ods html close;
ods listing;
```

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

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

ods html file='HTML-file-pathname.html';
ods pdf file='PDF-file-pathname.pdf';
ods rtf file='RTF-file-pathname.rtf';
ods ps file='PS-file-pathname.ps';
proc print data=employee_data(obs=12);
    id idnumber;
    title 'Personnel Data';
run;
ods _all_ close;
ods listing;

Note the two ODS statements that follow the PROC statement. The first one closes all files so that you can use them (for example, you could browse the HTML file or send the PDF file to a printer). The final statement opens the LISTING destination so that ODS returns to producing listing output for subsequent DATA or PROC steps in the current session.
Display 1.2  HTML 3.2 Output

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

Display 1.3  PDF Output

The following output is formatted in PDF and viewed with Adobe Acrobat Reader.
Display 1.4  RTF Output

The following RTF output is viewed with Microsoft Word 2000.

Display 1.5  PostScript Output

The following PostScript output is viewed with Ghostview.
Output 1.2  Listing Output

This output is traditional SAS listing output.

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

Where to Go from Here

- **Examples of ODS output:** To see the types of output that you can create with ODS, see “Gallery of ODS Samples” on page 14.
- **Essential concepts in ODS:** For concepts that will help you to understand and to use ODS to your best advantage, see “What Is the Output Delivery System?” on page 13.
- **Creating more complex HTML pages:** With ODS, you can create HTML pages that include a frame and a table of contents. For more information, see “ODS HTML Statement” on page 95 and Appendix 2, “ODS and the HTML Destination,” on page 637. You can see many examples of HTML output in *Base SAS Procedures Guide* online documentation.
- **ODS statements:** For reference information on the ODS statements, see Chapter 5, “Dictionary of ODS Language Statements,” on page 67. These statements control the many features of the Output Delivery System.
- **Using ODS with the DATA step:** With the addition of ODS-related options to the FILE and PUT statements, you can use ODS to produce enhanced DATA step reports. See Chapter 3, “Output Delivery System and the DATA Step,” on page 39.
- **Creating your own templates:** For even more control over formatting, you can create your own templates for formatting output. See Chapter 7, “TEMPLATE Procedure: Overview,” on page 261.
Concepts

Chapter 2. Output Delivery System: Basic Concepts 13
Chapter 3. Output Delivery System and the DATA Step 39
What Is the Output Delivery System?

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

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

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

**Gallery of ODS Samples**

**Introduction to the ODS Samples**

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

**Listing Output**

Traditional SAS output is Listing output. You do not need to change your SAS programs to create listing output. By default, you continue to create this kind of output even if you also create a type of output that contains more formatting.
### Average Quarterly Sales Amount by Each Sales Representative

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Garcia</th>
<th>Hollingsworth</th>
<th>Jensen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14752.5, 22806.1, 495.0, 63333.7</td>
<td>11926.9, 12165.2, 774.3, 31899.1</td>
<td>10015.7, 8009.5, 3406.7, 20904.8</td>
</tr>
<tr>
<td>2</td>
<td>18143.3, 20439.6, 1238.8, 53113.6</td>
<td>16026.8, 14355.0, 1237.5, 34686.4</td>
<td>12455.1, 12713.7, 1393.7, 34376.7</td>
</tr>
<tr>
<td>3</td>
<td>10729.8, 11457.0, 2787.3, 38712.5</td>
<td>7313.6, 7280.4, 1485.0, 30970.0</td>
<td>10585.3, 7361.7, 2227.5, 27129.7</td>
</tr>
<tr>
<td>4</td>
<td>11973.0, 10971.8, 3716.4, 30970.0</td>
<td>13624.4, 12624.6, 5419.8, 38093.1</td>
<td>19010.4, 15441.0, 1703.4, 38836.4</td>
</tr>
</tbody>
</table>

The MEANS Procedure

Analysis Variable: AmountSold
**PostScript Output**

With ODS, you can produce output in PostScript format.

**Display 2.1**  PostScript Output Viewed with Ghostview

---

**HTML Output**

With ODS, you can produce output in HTML (Hypertext Markup Language.) You can browse these files with Internet Explorer, Netscape, or any other browser that fully supports the HTML 3.2 tagset.

**Note:** To create HTML 4.0 tagsets, use the ODS HTML4 statement. In SAS 9, the ODS HTML statement generates HTML 3.2 tagsets. In future releases of SAS, the ODS HTML statement will support the most current HTML tagsets available. △
Display 2.2  HTML Output Viewed with Microsoft Internet Explorer

RTF Output

With ODS, you can produce RTF (Rich Text Format) output which is used with Microsoft Word.
With ODS, you can produce output in PDF (Portable Document Format), which can be viewed with the Adobe Acrobat Reader.
XML Output

With ODS, you can produce output that is tagged with XML (Extensible Markup Language) tags.
Output 2.2 XML Output file

```xml
<?xml version="1.0" encoding="windows-1252"?>
<odsxml>
<head>
<meta operator="user"/>
</head>
<body>
<proc name="Print">
<label name="IDX"/>
<title class="SystemTitle" toc-level="1">US Census of Population and Housing</title>
<branch name="Print" label="The Print Procedure" class="ContentProcName" toc-level="1">
<leaf name="Print" label="Data Set SASHELP.CLASS" class="ContentItem" toc-level="2">
<output name="Print" label="Data Set SASHELP.CLASS" clabel="Data Set SASHELP.CLASS">
<output-object type="table" class="Table">
<style>
<border spacing="1" padding="7" rules="groups" frame="box"/>
</style>
colspecs columns="6">
<colgroup>
<colspec name="1" width="2" align="right" type="int"/>
</colgroup>
<colgroup>
<colspec name="2" width="7" type="string"/>
<colspec name="3" width="1" type="string"/>
<colspec name="4" width="2" align="decimal" type="double"/>
<colspec name="5" width="4" align="decimal" type="double"/>
<colspec name="6" width="5" align="decimal" type="double"/>
</colgroup>
</colspecs>
<output-head>
<row>
<header type="string" class="Header" row="1" column="1">Obs</header>
<value type="double" class="RowHeader" row="2" column="1">1</value>
</row>
<row>
<header type="string" class="Header" row="1" column="2">Name</header>
<value type="string" class="Data" row="2" column="2">Alfred</value>
</row>
<row>
<header type="string" class="Header" row="1" column="3">Sex</header>
<value type="string" class="Data" row="2" column="3">M</value>
</row>
<row>
<header type="string" class="Header" row="1" column="4">Age</header>
<value type="string" class="Data" row="2" column="4">35</value>
</row>
<row>
<header type="string" class="Header" row="1" column="5">Height</header>
<value type="string" class="Data" row="2" column="5">68</value>
</row>
<row>
<header type="string" class="Header" row="1" column="6">Weight</header>
<value type="string" class="Data" row="2" column="6">150</value>
</row>
</output-head>
<output-body>
</body>
</odsxml>
```
Commonly Used ODS Terminology

data component
is a form, similar to a SAS data set, that contains the results (numbers and characters) of a DATA step or PROC step that supports ODS.

table definition
is a set of instructions that describes how to format the data. This description includes but is not limited to
- the order of the columns
- text and order of column headings
- formats for data
- font sizes and font faces.

output object
is an object that contains both the results of a DATA step or PROC step and information about how to format the results. An output object has a name, label, and path. For example, the Basic Statistical Measurement table generated from the UNIVARIATE procedure is an output object. It contains the data component and formatted presentation of the mean, median, mode, standard deviation, variance, range, and interquartile range.

Note: Although many output objects include formatting instructions, not all of them do. In some cases the output object consists of only the data component.

ODS destinations
are designations that produce specific types of output. ODS supports a number of destinations, including the following:

LISTING
produces traditional SAS output (monospace format).

Markup Languages
produce SAS output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX that you can access with a web browser. SAS supplies many markup languages for you to use ranging from DOCBOOK to TROFF. You can specify a markup language that SAS supplies or create one of your own and store it as a user-defined markup language.

DOCUMENT
produces a hierarchy of output objects that enables you to produce multiple ODS output formats without rerunning a PROC or DATA step and gives you more control over the structure of the output.

OUTPUT
produces a SAS data set.

Printer Family
produces output that is formatted for a high-resolution printer such as a PostScript (PS), PDF, or PCL file.

RTF
produces output that is formatted for use with Microsoft Word.
**ODS output**

ODS output consists of formatted output from any of the ODS destinations. For example, the OUTPUT destination produces SAS data sets; the LISTING destination produces listing output; the HTML destination produces output that is formatted in Hypertext Markup Language.

---

**How Does ODS Work?**

**Components of SAS Output**

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

The following figure shows how SAS produces ODS output.
Figure 2.1 ODS Processing: What Goes in and What Comes Out

ODS Processing: What Goes In and What Comes Out

Data Component + Table Definition

Output Object

DOCUMENT LISTING OUTPUT HTML MARKUP PRINTER RTF

SAS Formatted Destinations Third-Party Formatted Destinations

* List of Tagsets that SAS Supplies and Supports

Table 2.1 * List of Tagsets that SAS Supplies and Supports

<table>
<thead>
<tr>
<th>CHTML</th>
<th>HTML4</th>
<th>SASIOXML</th>
<th>SASXMOH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSVALL</td>
<td>HTMLCSS</td>
<td>SASREPORT</td>
<td>SASXMOIM</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>IMODE</td>
<td>SASXML</td>
<td>SASXMOR</td>
</tr>
<tr>
<td>DOCBOOK</td>
<td>PHTML</td>
<td>SASXMOG</td>
<td>WML</td>
</tr>
<tr>
<td>EVENT_MAP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* List of Tagsets that SAS Supplies but Does Not Support
Features of ODS

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

- ODS combines raw data with one or more table definitions to produce one or more output objects. These objects can be sent to any or all ODS destinations. You control the specific type of output from ODS by selecting an ODS destination. The currently available ODS destinations can produce:
  - traditional monospace output
  - an output data set
  - an ODS document that contains a hierarchy file of the output objects
  - output that is formatted for a high-resolution printer such as PostScript and PDF
  - output that is formatted in various markup languages such as HTML
  - RTF output that is formatted for use with Microsoft Word.

- ODS provides table definitions that define the structure of the output from SAS procedures and from the DATA step. You can customize the output by modifying these definitions, or by creating your own.

- ODS provides a way for you to choose individual output objects to send to ODS destinations. For example, PROC UNIVARIATE produces five output objects. You can easily create HTML output, an output data set, traditional listing output, or printer output from any or all of these output objects. You can send different output objects to different destinations.

- In the SAS windowing environment, ODS stores a link to each output object in the Results folder in the Results window.

- Because formatting is now centralized in ODS, the addition of a new ODS destination does not affect any procedures or the DATA step. As future destinations are added to ODS, they will automatically become available to the DATA step and all procedures that support ODS.

- With ODS, you can produce output for numerous destinations from a single source, but you do not need to maintain separate sources for each destination. This feature saves you time and system resources by enabling you to produce multiple kinds of output with a single run of your procedure or data query.

### Table 2.2  Additional Tagsets that SAS Supplies but Does Not Support

| COLORLATEX | LATEX | SHORT_MAP | TPL_STYLE_MAP |
| CSV | LATEX2 | STYLE_DISPLAY | TROFF |
| CSVBYLINE | NAMEDHTML | STYLE_POPUP | WMLOLIST |
| GRAPH | ODSSTYLE | TEXT_MAP |
| GTABLEAPPLET | PYX | TPL_STYLE_LIST |

**CAUTION:**

These tagsets are experimental tagsets. Do not use these tagsets in production jobs.
What Are the ODS Destinations?

Overview of ODS Destination Categories

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Destinations</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS Formatted</td>
<td>DOCUMENT</td>
<td>ODS document</td>
</tr>
<tr>
<td></td>
<td>LISTING</td>
<td>SAS output listing</td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
<td>SAS data set</td>
</tr>
<tr>
<td>Third-Party Formatted</td>
<td>HTML</td>
<td>HTML file for online viewing</td>
</tr>
<tr>
<td></td>
<td>MARKUP</td>
<td>markup language tagsets</td>
</tr>
<tr>
<td></td>
<td>PRINTER</td>
<td>printable output in one of three different formats: PCL, PDF, or PS (PostScript)</td>
</tr>
<tr>
<td></td>
<td>RTF</td>
<td>output written in Rich Text Format for use with Microsoft Word 2000</td>
</tr>
</tbody>
</table>

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

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

Definition of Destination-Independent Input

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

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

The three SAS formatted destinations are:

**DOCUMENT Destination**

The DOCUMENT destination enables you to restructure, navigate, and replay your data in different ways and to different destinations as you like without needing to rerun your analysis or repeat your database query. The DOCUMENT destination makes your entire output stream available in "raw" form and accessible to you to customize. The output is kept in the original internal representation as a data component plus a table definition. When the output is in a DOCUMENT form, it is possible to rearrange, restructure, and reformat without rerunning your analysis. Unlike other ODS destinations, the DOCUMENT destination has a GUI interface. However, everything that you can do through the GUI, you can also do with batch commands using the ODS DOCUMENT statement and the DOCUMENT procedure.

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

**LISTING Destination**

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

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

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

**OUTPUT Destination**

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

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

Note: For a list of style elements and valid values, see the style elements table in the SAS Output Delivery System: User's Guide.

The four categories of third-party formatted destinations are:

- **HTML (Hypertext Markup Language)**
  - The HTML destination produces HTML 3.2-compatible output. You can, however, produce (HTML 4 stylesheet) output using the HTML4 tagsets.
  - The HTML destination can create some or all of the following:
    - an HTML file (called the body file) that contains the results from the procedure
    - a table of contents that links to the body file
    - a table of pages that links to the body file
    - a frame that displays the table of contents, the table of pages, and the body file.
  - The body file is required with all ODS HTML output. If you do not want to link to your output, then you do not have to create a table of contents, a table of pages, or a frame file. However, if your output is very large, you might want to create a table of contents and a table of pages for easier reading and transversing through your file.
  - The HTML destination is intended only for on-line use, not for printing. To print hard-copies of the output objects, use the PRINTER destination.

- **Markup Languages (MARKUP) Family**
  - Just as table definitions describe how to lay out a table, and style attributes describe the style of the output, tagsets describe how to produce a markup language output. You can use a tagset that SAS supplies or you can create your own using the TEMPLATE procedure. Like table definitions and style attributes, tagsets enable you to modify your markup language output. For example, each variety of XML can be specified as a new tagset. SAS supplies you with a collection of XML tagsets and enables you to produce a customized variety of XML. The important point is that you can implement a tagset that SAS supplies or a customized tagset that you created without having to wait for the next release of SAS. With the addition of modifying and creating your own tagsets by using PROC TEMPLATE, now you have greater flexibility in customizing your output.
  - Because the MARKUP destination is so flexible, you can use either the SAS tagsets or a tagset that you created. For a complete listing of the markup language tagsets that SAS supplies, see the section on listing tagset names in the SAS Output Delivery System: User's Guide. To learn how to define your own tagsets, see the section on methods to create your own tagsets in the SAS Output Delivery System: User's Guide.
  - The MARKUP destination cannot replace ODS PRINTER or ODS RTF destinations because it cannot do text measurement. Therefore, it cannot produce output for a page description language or a hybrid language like RTF which requires all of the text to be measured and placed at a specific position on the page.
What Controls the Formatting Features of Third-Party Formats?

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

- ODS statement options
- ODS style attributes

The ODS statement options control three features:

1. Features that are specific to a given destination, such as stylesheets for HTML.
2. Features that are global to the document, such as AUTHOR and table of contents generation.
3. Features that we expect users to change on each document, such as the output file name.

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

**ODS Destinations and System Resources**

ODS destinations can be open or closed. You open and close a destination with the appropriate ODS statement. When a destination is open, ODS sends the output objects to it. An open destination uses system resources even if you use the selection and exclusion features of ODS to select or exclude all objects from the destination. Therefore, to conserve resources, close unnecessary destinations. For more information about using each destination, see the topic on ODS statements in the SAS Output Delivery System: User's Guide.

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

**What Are Table Definitions, Table Elements, and Table Attributes?**

A table definition describes how to generate the output for a tabular output object. (Most ODS output is tabular.) A table definition determines the order of column headers and the order of variables, as well the overall look of the output object that uses it. For information about customizing the table definition, see the topic on the TEMPLATE procedure in the SAS Output Delivery System: User's Guide.

In addition to the parts of the table definition that order the headers and columns, each table definition 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 definitions that control the presentation of the data have no effect on output objects that go to the LISTING or OUTPUT destination. However, the attributes that control the structure of the table and the data values do affect listing output.

For information on table attributes, see the section on table attributes in the SAS Output Delivery System: User's Guide.

**What Are Style Definitions, Style Elements, and Style Attributes?**

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

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

Each style attribute specifies a value for one aspect of the presentation. For example, the BACKGROUND= attribute specifies the color for the background of an HTML table or for a colored table in printed output. The FONT_STYLE= attribute specifies whether to use a Roman or an italic font. For information on style attributes, see the section on style attributes in the *SAS Output Delivery System: User’s Guide*.

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

---

**What Style Definitions Are Shipped with SAS Software?**

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

- **SAS Explorer Window:**
  To display a list of the available styles using the SAS Explorer Window, follow these steps:
  1. From any window in an interactive SAS session, select **View** ➤ **Results**
  2. In the Results window, select **View** ➤ **Templates**
  3. In the Templates window, select and open Sashelp.tmplmst.
  4. Select and open the **Styles** folder, which contains a list of available style definitions. If you want to view the underlying SAS code for a style definition, then select the style and open it.

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

- **TEMPLATE Procedure:**
  You can also display a list of the available styles by submitting the following PROC TEMPLATE statements:
  ```sas
  proc template;
  list styles;
  run;
  ```

- **SQL Procedure:**
  You can also display a list of the available styles by submitting the following PROC SQL statements:
  ```sas
  proc sql;
  select * from styles.style-name;
  ```

  The *style-name* is the name of any style from the template store (for example, *styles.default* or *styles.beige*).

  For more information on how ODS destinations use styles and how you can customize styles, see the section on the DEFINE STYLE statement in the *SAS Output Delivery System: User’s Guide*. 
How Do I Use Style Definitions with Base SAS Procedures?

- Most Base SAS Procedures
  
  Most Base SAS procedures that support ODS use one or more table definitions to produce output objects. These table definitions include definitions 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 custom styles for the ODS destinations that you use. For more information about customizing tables and styles, see the TEMPLATE procedure in the *SAS Output Delivery System: User’s Guide*.

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

Changing SAS Registry Settings for ODS

Overview of ODS and the SAS Registry

The SAS registry is the central storage area for configuration data that ODS uses. This configuration data is stored in a hierarchical form, which works in a similar manner to the way directory-based file structures work under UNIX, Windows, VMS, and the z/OS UNIX system. However, the SAS registry uses keys and subkeys as the basis for its structure, instead of using directories and subdirectories, like similar file systems in DOS or UNIX. A key is a word or a text string that refers to a particular aspect of SAS. Each key may be a place holder without values or subkeys associated with it, or it may have many subkeys with associated values. For example, the ODS key has DESTINATIONS, GUI, ICONS, and PREFERENCES subkeys. A subkey is a key inside another key. For example, PRINT is a subkey of the DESTINATIONS subkey.
Display 2.5  SAS Registry of ODS Subkeys

Changing Your Default HTML Version Setting

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

**CAUTION:**

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

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

1. Select
   - **Solutions** ➤ **Accessories** ➤ **Registry Editor**
   
   or
   
   Issue the command **REGEDIT**.

2. Select
   - **ODS** ➤ **Default HTML Version**

3. Select
   - **Edit** ➤ **Modify**
   
   or
   
   Click the right mouse button and select **MODIFY**. The Edit String Value window appears.

4. Type the HTML version in the **Value Data** text box and select **OK**.
Changing ODS Destination Default Settings

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

1. Select **ODS** ▶ Destinations
2. Select a destination subkey
3. Select a subkey in the Contents of window
4. Select **Edit** ▶ **Modify**

   *or*

   Click the right mouse button and select **MODIFY**.

5. Type in the Value Data entry into the Edit Value String or Edit Signed Integer Value window and select **OK**.
Customized ODS Output

SAS Output

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

Selection and Exclusion Lists

You can specify which output objects that you want to produce by selecting or excluding them in a list. For each ODS destination, ODS maintains either a selection list or an exclusion list. A selection list is a list of output objects that are sent to the destination. An exclusion list is a list of output objects that are excluded from the destination. ODS also maintains an overall selection list or an overall exclusion list. You can use these lists to control which output objects go to the specified ODS destinations.

To see the contents of the lists use the ODS SHOW statement. The lists are written to the SAS log. The following table shows the default lists:

Display 2.7  Registry Editor Window

![Registry Editor Window]
How Does ODS Determine the Destinations for an Output Object?

To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. For more information, about the ODS TRACE statement see SAS Output Delivery System: User's Guide. You can specify an output object as any of the following:

- a full path. For example,
  ```
  Univariate.City_Pop_90.TestsForLocation
  ```
  is the full path of the output object.

- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is
  ```
  Univariate.City_Pop_90.TestsForLocation
  ```
  then the partial paths are:
  ```
  City_Pop_90.TestsForLocation
  TestsForLocation
  ```

- a label that is enclosed in quotation marks.
  For example,
  ```
  "Tests For Location"
  ```

- a label path. For example, the label path for the output object is
  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```
  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement.

- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is
  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```
  then the partial label paths are:
  ```
  "CityPop_90"."Tests For Location"
  "Tests For Location"
  ```

- a mixture of labels and paths.

- any of the partial path specifications, followed by a pound sign (#) and a number. For example, TestsForLocation#3 refers to the third output object that is named TestsForLocation.
As each output object is produced, ODS uses the selection and exclusion lists to determine which destination or destinations the output object will be sent to. The following figure illustrates this process:

**Figure 2.2** Directing an Output Object to a Destination

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

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

**Customized Output for an Output Object**

For a procedure, the name of the table definition that is used for an output object comes from the procedure code. The DATA step uses a default table definition 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 the SAS Output Delivery System: User’s Guide.

To find out which table definitions 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:

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

Remember that not all procedures use table definitions. If you produce a trace record for one of these procedures, no definition appears in the trace record. Conversely, some procedures use multiple table definitions to produce their output. If you produce a trace record for one of these procedures, more than one definition appears in the trace record.
The trace record refers to the table definition as a template. For a detailed explanation of the trace record, see the section on the ODS TRACE statement in the *SAS Output Delivery System: User's Guide*.

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

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

**Summary of ODS**

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

Important features of ODS include the following:

- ODS combines raw data with one or more table definitions to produce one or more *output objects*. An output object tells ODS how to format the results of a procedure or DATA step.

- ODS provides table definitions that define the structure of the output from SAS procedures and from the DATA step. You can customize the output by modifying these definitions, or by creating your own definitions.

- ODS provides a way for you to choose individual output objects to send to ODS destinations.

- ODS stores a link to each output object in the Results folder for easy retrieval and access.

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

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

**DOCUMENT**

enables you to capture output objects from single run of the analysis and produce multiple reports in various formats whenever you want without re-running your SAS programs.

**LISTING**

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

**HTML**

produces output for online viewing.

**MARKUP**

produces output for markup language tagsets.
OUTPUT
produces SAS output data sets, thereby eliminating the need to parse PROC PRINTTO output.

PRINTER
produces presentation-ready printed reports.

RTF
produces output suitable for Microsoft Word reports.

By default, ODS output is formatted according to instructions that the procedure or DATA step defines. However, ODS provides ways for you to customize the presentation of your output. You can customize the presentation of your SAS output, or you can customize the look of a single output object. ODS gives you greater flexibility in generating, storing, and reproducing SAS procedure and DATA step output with a wide range of formatting options.
CHAPTER 3
Output Delivery System and the DATA Step

Why Use ODS with the DATA Step? 39
How ODS Works with the DATA Step 40
Syntax for ODS Enhanced Features in a DATA Step 41
Examples 41
  Example 1: Creating a Report with the DATA Step and the Default Table Definition 41
  Program 41
  Listing Output 44
  Example 2: Producing ODS Output That Contains Selected Variables 44
  Program 45
  HTML Output 47
  Listing Output 48
  Example 3: Assigning Attributes to Columns in ODS Output 48
  Program 48
  HTML Output 51
  Printer Output 52
  Listing Output 53
  Example 4: Creating and Using a User-Defined Table Definition Template 53
  Program: Creating the User-Defined Table Definition (Template) 54
  Program: Using the User-Defined Template (Table Definition) 54
  RTF Output 57

Why Use ODS with the DATA Step?

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

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

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

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

- the DATA _NULL_ statement
the FILE statement

the PUT statement

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

How ODS Works with the DATA Step

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

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

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

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

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

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

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

* If you do not specify a FILE statement, then the PUT statement writes to the SAS log by default. If you use multiple PUT and FILE statements, then in addition to creating ODS-enhanced output, you can write to the log, to the regular DATA step output buffer, or to another external file in the same DATA step.
Example 1: Creating a Report with the DATA Step and the Default Table Definition

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

<table>
<thead>
<tr>
<th>Topic</th>
<th>Where to learn more</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS formatting statements</td>
<td>Chapter 5, “Dictionary of ODS Language Statements,” on page 67</td>
</tr>
<tr>
<td>ODS destinations</td>
<td>“What Are the ODS Destinations?” on page 25</td>
</tr>
<tr>
<td>How ODS works</td>
<td>“How Does ODS Work?” on page 22</td>
</tr>
</tbody>
</table>

Syntax for ODS Enhanced Features in a DATA Step

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

FILE PRINT ODS<=(ODS-suboption(s))> <options>;
PUT <specification(s)> <_ODS_ <@|@@>> ;

Examples

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

ODS features:
FILE PRINT ODS statement:
PUT _ODS_ statement

ODS destinations:
LISTING

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

Program

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

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

Specify a title. The TITLE statement specifies a title for the output.
Example 1: Creating a Report with the DATA Step and the Default Table Definition

Chapter 3

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

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

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

```sas
data _null_;
```

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

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

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

```sas
file print ods;
```

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

```sas
put _ods_;
```

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

```sas
datalines;
  1995 BRZ Wheat 1516
  1995 BRZ Rice 11236
  1995 BRZ Corn 36276
  1995 CHN Wheat 102207
  1995 CHN Rice 185226
  1995 CHN Corn 112331
```
<table>
<thead>
<tr>
<th>Year</th>
<th>Country</th>
<th>Crop</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>IND</td>
<td>Wheat</td>
<td>63007</td>
</tr>
<tr>
<td>1995</td>
<td>IND</td>
<td>Rice</td>
<td>122372</td>
</tr>
<tr>
<td>1995</td>
<td>IND</td>
<td>Corn</td>
<td>9800</td>
</tr>
<tr>
<td>1995</td>
<td>INS</td>
<td>Wheat</td>
<td>.</td>
</tr>
<tr>
<td>1995</td>
<td>INS</td>
<td>Rice</td>
<td>49860</td>
</tr>
<tr>
<td>1995</td>
<td>INS</td>
<td>Corn</td>
<td>8223</td>
</tr>
<tr>
<td>1995</td>
<td>USA</td>
<td>Wheat</td>
<td>59494</td>
</tr>
<tr>
<td>1995</td>
<td>USA</td>
<td>Rice</td>
<td>7888</td>
</tr>
<tr>
<td>1995</td>
<td>USA</td>
<td>Corn</td>
<td>187300</td>
</tr>
<tr>
<td>1996</td>
<td>BRZ</td>
<td>Wheat</td>
<td>3302</td>
</tr>
<tr>
<td>1996</td>
<td>BRZ</td>
<td>Rice</td>
<td>10035</td>
</tr>
<tr>
<td>1996</td>
<td>BRZ</td>
<td>Corn</td>
<td>31975</td>
</tr>
<tr>
<td>1996</td>
<td>CHN</td>
<td>Wheat</td>
<td>109000</td>
</tr>
<tr>
<td>1996</td>
<td>CHN</td>
<td>Rice</td>
<td>190100</td>
</tr>
<tr>
<td>1996</td>
<td>CHN</td>
<td>Corn</td>
<td>119350</td>
</tr>
<tr>
<td>1996</td>
<td>IND</td>
<td>Wheat</td>
<td>62620</td>
</tr>
<tr>
<td>1996</td>
<td>IND</td>
<td>Rice</td>
<td>120012</td>
</tr>
<tr>
<td>1996</td>
<td>IND</td>
<td>Corn</td>
<td>8660</td>
</tr>
<tr>
<td>1996</td>
<td>INS</td>
<td>Wheat</td>
<td>.</td>
</tr>
<tr>
<td>1996</td>
<td>INS</td>
<td>Rice</td>
<td>51165</td>
</tr>
<tr>
<td>1996</td>
<td>INS</td>
<td>Corn</td>
<td>8925</td>
</tr>
<tr>
<td>1996</td>
<td>USA</td>
<td>Wheat</td>
<td>62099</td>
</tr>
<tr>
<td>1996</td>
<td>USA</td>
<td>Rice</td>
<td>7771</td>
</tr>
<tr>
<td>1996</td>
<td>USA</td>
<td>Corn</td>
<td>236064</td>
</tr>
</tbody>
</table>

;
Example 2: Producing ODS Output That Contains Selected Variables

ODS features:

FILE PRINT ODS statement:
VARIABLES= suboption

ODS HTML statement:
BODY= option
    URL= suboption
    PUT _ODS_ statement

ODS destinations:
    HTML
    LISTING

---

Listing Output

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

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

<table>
<thead>
<tr>
<th>Country</th>
<th>Leading Grain Producers</th>
<th>Year</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Wheat</td>
<td>1995</td>
<td>1516</td>
</tr>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>1995</td>
<td>11236</td>
</tr>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>1995</td>
<td>36276</td>
</tr>
<tr>
<td>China</td>
<td>Wheat</td>
<td>1995</td>
<td>102207</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>1995</td>
<td>185226</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>1995</td>
<td>112331</td>
</tr>
<tr>
<td>India</td>
<td>Wheat</td>
<td>1995</td>
<td>63007</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>1995</td>
<td>122372</td>
</tr>
<tr>
<td>India</td>
<td>Corn</td>
<td>1995</td>
<td>9800</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Wheat</td>
<td>1995</td>
<td>.</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>1995</td>
<td>49860</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Corn</td>
<td>1995</td>
<td>8223</td>
</tr>
<tr>
<td>United States</td>
<td>Wheat</td>
<td>1995</td>
<td>59494</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>1995</td>
<td>7888</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>1995</td>
<td>187300</td>
</tr>
<tr>
<td>Brazil</td>
<td>Wheat</td>
<td>1996</td>
<td>3302</td>
</tr>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>1996</td>
<td>10035</td>
</tr>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>1996</td>
<td>31975</td>
</tr>
<tr>
<td>China</td>
<td>Wheat</td>
<td>1996</td>
<td>109000</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>1996</td>
<td>190100</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>1996</td>
<td>119350</td>
</tr>
<tr>
<td>India</td>
<td>Wheat</td>
<td>1996</td>
<td>62620</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>1996</td>
<td>120012</td>
</tr>
<tr>
<td>India</td>
<td>Corn</td>
<td>1996</td>
<td>8660</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Wheat</td>
<td>1996</td>
<td>.</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>1996</td>
<td>51165</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Corn</td>
<td>1996</td>
<td>8925</td>
</tr>
<tr>
<td>United States</td>
<td>Wheat</td>
<td>1996</td>
<td>62099</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>1996</td>
<td>7771</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>1996</td>
<td>236064</td>
</tr>
</tbody>
</table>
Example 2: Producing ODS Output That Contains Selected Variables

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

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program

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

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

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

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

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

```sas
title ‘Leading Grain Producers’;
title2 ‘for 1996’;
```

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

```sas
data _null_;
```

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

```sas
length Country $ 3 Type $ 5;
format country $cntry.;
label type='Grain';
Read a record from the input data, assign values to four variables. Continue to process only observations that meet the criterion. The INPUT statement reads a single record and assigns values to four variables. The subsetting IF statement causes the DATA step to continue to process only those observations that contain the value 1996 for YEAR.

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

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

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

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

```plaintext
    put _ods_;
```

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

```plaintext
data lines;
1995 BRZ Wheat 1516
1995 BRZ Rice 11236
1995 BRZ Corn 36276
1995 CHN Wheat 102207
1995 CHN Rice 185226
1995 CHN Corn 112331
1995 IND Wheat 63007
1995 IND Rice 122372
1995 IND Corn 9800
1995 INS Wheat .
1995 INS Rice 49860
1995 INS Corn 8223
1995 USA Wheat 59494
1995 USA Rice 7888
1995 USA Corn 187300
1996 BRZ Wheat 3302
1996 BRZ Rice 10035
1996 BRZ Corn 31975
1996 CHN Wheat 109000
1996 CHN Rice 190100
1996 CHN Corn 119350
1996 IND Wheat 62620
1996 IND Rice 120012
1996 IND Corn 8660
```
Example 2: Producing ODS Output That Contains Selected Variables

1996 INS Wheat .
1996 INS Rice 51165
1996 INS Corn 8925
1996 USA Wheat 62099
1996 USA Rice 7771
1996 USA Corn 236064
;

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

ods html close;

HTML Output

Display 3.1  HTML Body File Produced by ODS

<table>
<thead>
<tr>
<th>Country</th>
<th>Grain</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Wheat</td>
<td>3302</td>
</tr>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10035</td>
</tr>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>31975</td>
</tr>
<tr>
<td>China</td>
<td>Wheat</td>
<td>109000</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190100</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>119350</td>
</tr>
<tr>
<td>India</td>
<td>Wheat</td>
<td>62620</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>120012</td>
</tr>
<tr>
<td>India</td>
<td>Corn</td>
<td>8660</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Wheat</td>
<td>51165</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>8925</td>
</tr>
<tr>
<td>United States</td>
<td>Wheat</td>
<td>62099</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7771</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>236064</td>
</tr>
</tbody>
</table>
### Listing Output

#### Output 3.2  Listing Output Produced by the LISTING Destination

<table>
<thead>
<tr>
<th>Country</th>
<th>Grain</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Wheat</td>
<td>3302</td>
</tr>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10035</td>
</tr>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>31975</td>
</tr>
<tr>
<td>China</td>
<td>Wheat</td>
<td>109000</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190100</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>119350</td>
</tr>
<tr>
<td>India</td>
<td>Wheat</td>
<td>62620</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>120012</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Corn</td>
<td>8660</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Wheat</td>
<td></td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>51165</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Corn</td>
<td>8925</td>
</tr>
<tr>
<td>United States</td>
<td>Wheat</td>
<td>62099</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7771</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>236064</td>
</tr>
</tbody>
</table>

---

### Example 3: Assigning Attributes to Columns in ODS Output

ODS features:

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

ODS destinations:

- **HTML**
- **Listing**
- **Printer (PS)**

**Format:**

- `$CNTRY. on page 42`

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

**Note:** This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649. ⬤

### Program

**Set the SAS system options.** The NODATE option suppresses the display of the date and time in the output. The PAGENO= option specifies the starting page number. The LINESIZE= option specifies the output line length, and the PAGESIZE= option specifies the number of lines on an output page. These options affect the listing output, but none of them affects the HTML output.
Example 3: Assigning Attributes to Columns in ODS Output

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

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

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

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

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

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

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

```article
data _null_;
```

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

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

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

```article
input Year country $ type $ Kilotons;
if year=1996;
```
Send the DATA step output to the open destinations, specify a label for the output object, and specify the variables to write to the data component and the order in which to write them. The combination of the fileref PRINT and the ODS option in the FILE statement sends the results of the DATA step to ODS. The LISTING, the HTML, and the PRINTER destinations are open. Because no table definition is specified, ODS uses the default DATA step definition.

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

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

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

```plaintext
put _ods_
;
```

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

```plaintext
datalines;
  1995 BRZ Wheat 1516
  1995 BRZ Rice 11236
  1995 BRZ Corn 36276
  1995 CHN Wheat 102207
  1995 CHN Rice 185226
  1995 CHN Corn 112331
  1995 IND Wheat 63007
  1995 IND Rice 122372
  1995 IND Corn 9800
  1995 INS Wheat .
  1995 INS Rice 49860
  1995 INS Corn 8223
  1995 USA Wheat 59494
  1995 USA Rice 7888
  1995 USA Corn 187300
  1996 BRZ Wheat 3302
  1996 BRZ Rice 10035
  1996 BRZ Corn 31975
  1996 CHN Wheat 109000
  1996 CHN Rice 190100
  1996 CHN Corn 119350
  1996 IND Wheat 62620
```
To view the HTML output and print the PostScript output, close both the HTML and PRINTER destinations. This statement closes the LISTING, HTML and PRINTER destinations and all the files that are associated with them. You must close the HTML destination before you can view the output with a browser. You must close the PRINTER destination before you can print the output on a physical printer. If you do not close these destinations, then output created in subsequent sessions will be routed to them, and you might inadvertently continue to generate both HTML and PostScript output.

ods _all_ close;

HTML Output

Display 3.2 HTML Frame File Produced by ODS

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

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

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

Display 3.3  Printer Output Viewed with Ghostview

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

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

### Leading Grain Producers for 1996

<table>
<thead>
<tr>
<th>Country</th>
<th>Type of Grain</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Wheat</td>
<td>3,302</td>
</tr>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10,035</td>
</tr>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>31,975</td>
</tr>
<tr>
<td>China</td>
<td>Wheat</td>
<td>109,000</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190,100</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>119,350</td>
</tr>
<tr>
<td>India</td>
<td>Wheat</td>
<td>62,620</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>123,012</td>
</tr>
<tr>
<td>India</td>
<td>Corn</td>
<td>8,660</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Wheat</td>
<td></td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>51,165</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Corn</td>
<td>8,925</td>
</tr>
<tr>
<td>United States</td>
<td>Wheat</td>
<td>62,099</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7,771</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>236,064</td>
</tr>
</tbody>
</table>
Listing Output

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

<table>
<thead>
<tr>
<th>Country</th>
<th>Type</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Wheat</td>
<td>3,302</td>
</tr>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10,035</td>
</tr>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>31,975</td>
</tr>
<tr>
<td>China</td>
<td>Wheat</td>
<td>109,000</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190,100</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>119,350</td>
</tr>
<tr>
<td>India</td>
<td>Wheat</td>
<td>62,620</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>120,012</td>
</tr>
<tr>
<td>India</td>
<td>Corn</td>
<td>8,660</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Wheat</td>
<td>51,165</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>8,925</td>
</tr>
<tr>
<td>United States</td>
<td>Wheat</td>
<td>62,099</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7,771</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>236,064</td>
</tr>
</tbody>
</table>

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

ODS features:

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

ODS destination:
  RTF
```

This example shows how to:
- create a simple user-defined template (table definition) with PROC TEMPLATE
- use a simple user-defined template in the DATA step
- use pointer controls in the PUT _ODS_ statement.
Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

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

Define the table definition PHONELIST. This PROC TEMPLATE step defines a table definition named PHONELIST.
The template defines two columns: NAME and PHONE.
The GENERIC=ON attribute defines the column for NAME as one that the DATA step can use for multiple variables.
The column definition uses dynamic headers; that is, a variable that uses this column definition takes the value of the header at run time from the DATA step that uses this template. Thus, each variable can have a different column header.
The STYLE= attribute specifies that the style element DATA be used as the basis for generating the data in this column. The font face and font size that DATA normally uses are replaced by the ones that are specified in the STYLE= attribute.
The header for PHONE is hard-coded as Telephone. The STYLE= attribute specifies a style element to use for the data in this column. For information on PROC TEMPLATE, see Chapter 7, “TEMPLATE Procedure: Overview,” on page 261.

```
proc template;
define table phonelist;
   column name phone;
dynamic colheader;
define name;
generic=on;
   header=colheader;
   style=data{font_style=italic font_size=5};
end;

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

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

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

```
ods listing close;
```

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

```
ods rtf body='your_rtf_file.rtf';
```
Specify a title. The TITLE statement provides a title for the output.

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

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

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

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

```
data phones;
    length first_name $20 last_name $25;
    input first_name $ last_name $ business_phone home_phone;

datalines;
Jerome Johnson 9193191677 9198462198
Romeo Montague 8008992164 3609736201
Imani Rashid 5088522146 5083669821
Falnor Kent . 9197823199
Ruby Archuleta . .
Takei Ito 7042982145 .
Tom Joad 2099632764 2096684741
;
```

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

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

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

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

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

```
file print ods=(template='phonelist'
```
Place variable values in columns. The COLUMNS= suboption places values of variables into columns that are defined in the template.

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

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

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

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

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

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

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

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

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

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

```plaintext
ods RTF close;
```
RTF Output

Display 3.4  RTF Output Viewed with Microsoft Word
PART 3

ODS Language Statements

Chapter 4 . . . . . . . Introduction to ODS Language Statements  61
Chapter 5 . . . . . . . Dictionary of ODS Language Statements  67
CHAPTER

Introduction to ODS Language Statements

Definition of ODS Statements

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

Types of ODS Statements

DATA Step Statements

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

Global Statements

- provide information to SAS
- request information or data
- move between different modes of execution
- set values for system options.
The global ODS statements deliver or store output in a variety of formats. You can use global statements anywhere in a SAS program. Global Statements are not executable; they take effect as soon as SAS compiles program statements. Global ODS statements are organized into three categories:

**ODS: Output Control**

Statements that provide descriptive information about the specified output objects and indicate whether or not the style definition or table definition is supplied by SAS. The Output Control statements can do the following:

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

**ODS: SAS Formatted**

Statements that enable you to produce SAS specific items such as a SAS data set, SAS output listing, or an ODS document. The statements in the ODS SAS Formatted category create the SAS entities. For more information, see “The SAS Formatted Destinations” on page 26.

**ODS: Third-Party Formatted**

Statements that enable you to apply styles and markup languages, or produce output to physical printers, using page description languages. For more information, see “The Third-Party Formatted Destinations” on page 27.

---

## Procedure Statements

For information about the TEMPLATE procedure, see Chapter 7, “TEMPLATE Procedure: Overview,” on page 261. For information about the DOCUMENT procedure, see Chapter 6, “The DOCUMENT Procedure,” on page 211.

---

## ODS Statement Category Descriptions

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

<table>
<thead>
<tr>
<th>Statement category</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS: Output Control</td>
<td>Provide descriptive information about the specified output objects and their locations.</td>
</tr>
<tr>
<td>ODS: SAS Formatted</td>
<td>Produce listing output, a SAS output data set, or a hierarchy file.</td>
</tr>
<tr>
<td>ODS: Third-party Formatted</td>
<td>Produce files that are formatted in the proper destination format.</td>
</tr>
</tbody>
</table>
# ODS Statements by Category

## Table 4.2  Categories and Descriptions of ODS Statements

<table>
<thead>
<tr>
<th>Category</th>
<th>Dictionary of ODS Language Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File-handling</td>
<td>“FILE Statement for ODS” on page 68</td>
<td>Creates an ODS output object by binding the data component to the table definition (template). Optionally, lists the variables to include in the ODS output, and specifies options that control the way that the variables are formatted.</td>
</tr>
<tr>
<td></td>
<td>“PUT Statement for ODS” on page 80</td>
<td>Writes data values to a special buffer from which they can be written to the data component and then formatted by ODS.</td>
</tr>
<tr>
<td>ODS: Output Control</td>
<td>“LIBNAME Statement, SASEDOC” on page 76</td>
<td>Uses the SASEDOC engine to associate a SAS libref (library reference) with one or more ODS output objects that are stored in an ODS document.</td>
</tr>
<tr>
<td></td>
<td>“ODS <em>ALL</em> CLOSE Statement” on page 84</td>
<td>Closes all open ODS output destinations</td>
</tr>
<tr>
<td></td>
<td>“ODS DOCUMENT Statement” on page 87</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“ODS EXCLUDE Statement” on page 90</td>
<td>Specifies output objects to exclude from ODS destinations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enables ODS automatic graphic capabilities</td>
</tr>
<tr>
<td></td>
<td>“ODS PATH Statement” on page 149</td>
<td>Specifies locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them.</td>
</tr>
<tr>
<td></td>
<td>“ODS PROCLABEL Statement” on page 175</td>
<td>Enables you to change a procedure label</td>
</tr>
<tr>
<td></td>
<td>“ODS PROCTITLE Statement” on page 176</td>
<td>Determines whether or not to write the title that identifies the procedure that produces the results in the output</td>
</tr>
<tr>
<td></td>
<td>“ODS RESULTS Statement” on page 179</td>
<td>Tracks ODS output in the Results window</td>
</tr>
<tr>
<td></td>
<td>“ODS SELECT Statement” on page 188</td>
<td>Specifies output objects for ODS destinations</td>
</tr>
<tr>
<td></td>
<td>“ODS SHOW Statement” on page 197</td>
<td>Writes the specified selection or exclusion list to the SAS log</td>
</tr>
<tr>
<td></td>
<td>“ODS TRACE Statement” on page 197</td>
<td>Writes to the SAS log a record of each output object that is created, or suppresses the writing of this record</td>
</tr>
<tr>
<td></td>
<td>“ODS USEGOPT Statement” on page 202</td>
<td>Determines whether or or not ODS uses graphics option settings</td>
</tr>
<tr>
<td>Category</td>
<td>Dictionary of ODS Language Statement</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ODS: SAS Formatted</td>
<td>“ODS VERIFY Statement” on page 205</td>
<td>Prints or suppresses a message indicating that a style definition or a table definition being used is not supplied by SAS</td>
</tr>
<tr>
<td></td>
<td>“ODS DECIMAL_ALIGN Statement” on page 85</td>
<td>Controls the justification of numeric columns when no justification is specified</td>
</tr>
<tr>
<td></td>
<td>“ODS LISTING Statement” on page 107</td>
<td>Opens, manages, or closes the LISTING destination</td>
</tr>
<tr>
<td></td>
<td>“ODS OUTPUT Statement” on page 135</td>
<td>Produces a SAS data set from an output object and manages the selection and exclusion lists for the OUTPUT destination</td>
</tr>
<tr>
<td>ODS: Third-Party Formatted</td>
<td>“ODS CHTML Statement” on page 84</td>
<td>Opens, manages, or closes the CHTML destination, which produces a compact, minimal HTML that does not use style information</td>
</tr>
<tr>
<td></td>
<td>“ODS CSVALL Statement” on page 85</td>
<td>Opens, manages, or closes the CSVALL destination, which produces output containing columns of data values that are separated by commas, and produces tabular output with titles, notes, and bylines</td>
</tr>
<tr>
<td></td>
<td>“ODS DOCBOOK Statement” on page 86</td>
<td>Opens, manages, or closes the DOCBOOK destination, which produces XML output that conforms to the DocBook DTD by OASIS</td>
</tr>
<tr>
<td></td>
<td>“ODS HTML Statement” on page 95</td>
<td>Opens, manages, or closes the HTML destination, which produces HTML 4.0 output that contains embedded stylesheets</td>
</tr>
<tr>
<td></td>
<td>“ODS HTMLCSS Statement” on page 105</td>
<td>Opens, manages, or closes the HTMLCSS destination, which produces HTML output with cascading style sheets</td>
</tr>
<tr>
<td></td>
<td>“ODS HTML3 Statement” on page 105</td>
<td>Opens, manages, or closes the HTML3 destination, which produces HTML 3.2 formatted output</td>
</tr>
<tr>
<td></td>
<td>“ODS IMODE Statement” on page 106</td>
<td>Opens, manages, or closes the IMODE destination, which produces HTML output as a column of output, separated by lines</td>
</tr>
<tr>
<td></td>
<td>“ODS MARKUP Statement” on page 109</td>
<td>Opens, manages, or closes the MARKUP destination, which produces SAS output that is formatted using one of many different markup languages</td>
</tr>
<tr>
<td></td>
<td>“ODS PCL Statement” on page 150</td>
<td>Opens, manages, or closes the PCL destination, which produces printable output for PCL (HP LaserJet) files</td>
</tr>
<tr>
<td></td>
<td>“ODS PDF Statement” on page 153</td>
<td>Opens, manages, or closes the PDF destination, which produces PDF output, a form of output that is read by Adobe Acrobat Reader and other applications</td>
</tr>
<tr>
<td></td>
<td>“ODS PHTML Statement” on page 159</td>
<td>Opens, manages, or closes the PHTML destination, which produces simple HTML output that uses twelve style elements and no class attributes</td>
</tr>
<tr>
<td></td>
<td>“ODS PRINTER Statement” on page 159</td>
<td>Opens, manages, or closes the PRINTER destination, which produces printable output</td>
</tr>
<tr>
<td></td>
<td>“ODS PS Statement” on page 177</td>
<td>Opens, manages, or closes the PS destination, which Produces PostScript (PS) output</td>
</tr>
<tr>
<td>Category</td>
<td>Dictionary of ODS Language Statement</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>“ODS RTF Statement” on page 180</td>
<td>Opens, manages, or closes the RTF destination, which produces output written in Rich Text Format for use with Microsoft Word 2000</td>
</tr>
<tr>
<td></td>
<td>“ODS WML Statement” on page 206</td>
<td>Opens, manages, or closes the WML destination, which uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a simple list for a table of contents</td>
</tr>
</tbody>
</table>
CHAPTER 5

Dictionary of ODS Language Statements

FILE Statement for ODS 68
LIBNAME Statement, SASEDOC 76
PUT Statement for ODS 80
ODS _ALL_ CLOSE Statement 84
ODS CHTML Statement 84
ODS CVSALL Statement 85
ODS DECIMAL ALIGN Statement 85
ODS DOCBOOK Statement 86
ODS DOCUMENT Statement 87
ODS EXCLUDE Statement 90
ODS GRAPHICS Statement (Experimental) 92
ODS HTML Statement 95
ODS HTMLCSS Statement 105
ODS HTML3 Statement 105
ODS IMODE Statement 106
ODS LISTING Statement 107
ODS MARKUP Statement 109
   Specifying a Tagset Keyword as an ODS Destination 126
   Specifying a Two-Level Tagset Name as an ODS Destination 126
ODS OUTPUT Statement 135
ODS PATH Statement 149
ODS PCL Statement 150
ODS PDF Statement 153
ODS PHTML Statement 159
ODS PRINTER Statement 159
ODS PROCLABEL Statement 175
ODS PROCTITLE Statement 176
ODS PS Statement 177
ODS RESULTS Statement 179
ODS RTF Statement 180
ODS SELECT Statement 188
ODS SHOW Statement 197
ODS TRACE Statement 197
ODS USEGOPT Statement 202
ODS VERIFY Statement 205
ODS WML Statement 206
FILE Statement for ODS

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

Valid: in a DATA step
Category: File-handling
Type: Executable
Default: ODS sends the output object to all open ODS destinations.

Syntax

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

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

Required Arguments

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

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

Featured in: “Example 1: Creating a Report with the DATA Step and the Default Table Definition” in SAS Output Delivery System: User’s Guide

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

See Also: For information about the ODS suboptions, see “ODS Suboptions” on page 69.

Featured in: All examples

Options

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

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

DROPOVER
discards items when a PUT statement attempts to write beyond the last ODS column in the buffer. A message in the log at the end of the DATA step informs you if data were not written to the buffer.
FLOWOVER
moves the output pointer to a new line if a PUT statement attempts to write an item beyond the last ODS column in the buffer. The PUT statement writes the next item in the first ODS column of the new line.

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

Default: FLOWOVER

Without ODS Suboptions
If you do not specify any ODS suboptions, then the DATA step uses a default table definition (BASE.DATASTEP.TABLE) that is stored in the SASHELP.TMPLMST template store. This definition defines two generic columns: one for character variables and one for numeric variables. ODS associates each variable in the DATA step with one of these columns and displays the variables in the order in which they are defined in the DATA step.

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

**ODS Suboptions**

<table>
<thead>
<tr>
<th>Task</th>
<th>Suboption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify one or more columns for the data component</td>
<td>COLUMNS= or VARIABLES=</td>
</tr>
<tr>
<td>Specify default values for column attributes that exist in the table definition, but that get their values from the data component</td>
<td>DYNAMIC=</td>
</tr>
<tr>
<td>Specify whether all column definitions in the table definition can be used by more than one variable</td>
<td>GENERIC=</td>
</tr>
<tr>
<td>Specify a column header to use for any column that does not have a column header specified in the COLUMNS= or VARIABLES= suboption</td>
<td>LABEL=</td>
</tr>
<tr>
<td>Specify a name for the output object that the DATA step produces</td>
<td>OBJECT=</td>
</tr>
<tr>
<td>Specify a label for the output object that the DATA step produces</td>
<td>OBJECTLABEL=</td>
</tr>
<tr>
<td>Specify the table definition to use with the data component to produce the output object</td>
<td>TEMPLATE=</td>
</tr>
</tbody>
</table>
COLUMN=\textit{\texttt{column-specification(s)}}

specifies one or more columns for the data component and determines their order in
the data component.

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

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

\textbf{Requirement:} You must enclose \textit{\texttt{column-specification(s)}} in parentheses.

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

\textbf{Tip:} You can override the default order by using the ORDER_DATA= table attribute
in the PROC TEMPLATE step that creates the definition. The default DATA step
table definition uses this attribute. For more information see the discussion of
ORDER_DATA= on page 419 table attribute.

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

Each \textit{\texttt{column-specification}} associates a DATA step variable with a column that is
deefined in the table definition. \textit{\texttt{column-specification}} has this general form:

\begin{verbatim}
\begin{align*}
\begin{array}{ll}
\text{column-name-1} & \leq \text{variable-name-1} \langle \text{attribute-suboptions} \rangle \\
\text{column-name-n} & \leq \text{variable-name-n} \langle \text{attribute-suboptions} \rangle
\end{array}
\end{align*}
\end{verbatim}

\textit{\texttt{column-name}}

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

\textbf{Restriction:} \textit{\texttt{column-name}} must conform to the rules for SAS variable names. For
information see the SAS Language Reference: Dictionary.

\textbf{Requirement:} You must enclose \textit{\texttt{column-name}} in parentheses.

\textbf{Tip:} You can use list notation (for example, \texttt{score1-score5}) to specify multiple
\textit{\texttt{column-name}} names.

\textbf{Featured in:} “Example 4: Creating and Using a User-Defined Table Definition
Template” in SAS Output Delivery System: User’s Guide

\textit{\texttt{variable-name}}

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

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

\textbf{Tip:} You can use list notation (for example, \texttt{score1-score5}) to specify a range of
variable names.

\textbf{Featured in:} “Example 4: Creating and Using a User-Defined Table Definition
Template” in SAS Output Delivery System: User’s Guide
(attribute-suboptions) assigns a characteristic, such as a label or a format, to a particular column in the data component. These individual specifications override any attributes that is set by the DATA step.

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Suboption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a value for a column attribute that exists in the table definition, but that gets its value from the data component</td>
<td>DYNAMIC=</td>
</tr>
<tr>
<td>Specify a format for the current column</td>
<td>FORMAT=</td>
</tr>
<tr>
<td>Specify whether the DATA step uses this column definition for multiple variables</td>
<td>GENERIC=</td>
</tr>
<tr>
<td>Specify a label for a particular column</td>
<td>LABEL=</td>
</tr>
</tbody>
</table>

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

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

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

Each dynamic-specification has the following form:

\[\text{dynamic-value-name} = \text{variable-name} \mid \text{constant}\]

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

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

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

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

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

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

### GENERIC=ON | OFF

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

- **ON** indicates whether the DATA step uses all column definitions for multiple variables.
- **OFF** indicates whether the DATA step uses no column definitions for multiple variables.

**Default:** OFF

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

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

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

### LABEL='column-label'

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

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

- a label that is specified with HEADER= attribute for a particular column in the table definition (see HEADER= on page 382 column attribute).
- a label that is specified for a particular column with LABEL= suboption in the COLUMNS= or VARIABLES= suboption.
- a label that is specified with LABEL= suboption in the ODS= option.
- a label that is assigned with the LABEL statement in the DATA step.

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

**Featured in:** “Example 3: Assigning Attributes to Columns in ODS Output” in SAS Output Delivery System: User’s Guide

### OBJECT= object-name

specifies a name for the output object.

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

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

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

### OBJECTLABEL='object-label'

specifies a label for the output object.

The Results window and the HTML contents file both contain a description of, and a link to, each output object. The description contains the first of the following items that ODS finds:
the object’s label
the current title if it is not the default title, “The SAS System”
the object’s name (see OBJECT= on page 72)
the string FilePrint#, where # increases by 1 for each DATA step that you run in the current SAS process without specifying an object name or an object label.

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

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

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

Default: If you do not specify the TEMPLATE= option, ODS uses BASE.DATASTEP.TABLE, the default table definition.
Default: If you do specify the TEMPLATE= suboption, ODS first looks for table-definition-name in SASUSER.TEMPLAT, and then it looks in SASHELP.TMPLMST.

Requirement: You must enclose table-definition-name in quotation marks.
Interaction: When you use the default table definition, the GENERIC= suboption is set to ON for all columns in the data component. For more information see GENERIC= on page 72.
Tip: When you use the BASE.DATASTEP.TABLE template, character values are left-justified. If you want character values to be right-justified, specify the BASE.DATASTEP.TABLENOJUST template.
Tip: You can change the locations that ODS searches for the table-definition-name by using the “ODS PATH Statement” on page 149.

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

  (variable-name-1<=column-name-1<(attribute-suboptions)>> < . . . variable-name-n<=column-name-n<(attribute-suboptions)>>)
**Default:** If you are using the default table definition and you omit `column-name`, then ODS uses the variable label to name the column. If the variable has no label, then ODS uses the variable name.

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

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

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

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

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Suboption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a value for a column attribute that exists in the table definition, but that gets its value from the data component</td>
<td>DYNAMIC=</td>
</tr>
<tr>
<td>Specify a format for the current column</td>
<td>FORMAT=</td>
</tr>
<tr>
<td>Specify whether the DATA step uses this column definition for multiple variables</td>
<td>GENERIC=</td>
</tr>
<tr>
<td>Specify a label for a particular column</td>
<td>LABEL=</td>
</tr>
</tbody>
</table>

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

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

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

**Tip:** You can override the default order by using the ORDER_DATA table attribute in the PROC TEMPLATE step that creates the definition. The default DATA step table definition uses this attribute. For more information see the ORDER_DATA= table attribute.

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

**Attribute Suboptions**

**DYNAMIC=** *dynamic-specification(s)*

specifies a value for a column attribute that exists in the table definition, but that get its value from the data component.

**Main discussion:** DYNAMIC= on page 71

**Featured in:** “Example 4: Creating and Using a User-Defined Table Definition Template” in *SAS Output Delivery System: User’s Guide*

**FORMAT=** *format-name*

specifies a format for the current column.

**Default:** ODS uses the first of these formats for the variable that it finds:
- for nongeneric columns, a format that is specified in the column definition
- a format that is specified in the FORMAT= column attribute
- a format that is specified in a FORMAT statement
- the default format ($w$. for character variables; BEST12. for numeric variables).

**Featured in:** “Example 4: Creating and Using a User-Defined Table Definition Template” in *SAS Output Delivery System: User’s Guide*

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

**GENERIC=** *ON|OFF*

specifies whether the DATA step uses this column definition for multiple variables.

**Default:** OFF

**Main discussion:** GENERIC= on page 72

**Featured in:** “Example 4: Creating and Using a User-Defined Table Definition Template” in *SAS Output Delivery System: User’s Guide*

**LABEL=** *column-label*

specifies a label for the specified column.

**Main discussion:** LABEL= on page 72

**Featured in:** “Example 3: Assigning Attributes to Columns in ODS” in *SAS Output Delivery System: User’s Guide*

**Details**

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

- These arguments affect only listing output:
  - FOOTNOTES and NOFOOTNOTES
  - LINESIZE
  - PAGESIZE
  - TITLE and NOTITLES

- Do not use these arguments:
  - DELIMITER=}
LIBNAME Statement, SASEDOC

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

Valid: Anywhere
Category: ODS: Output Control
Restriction: The LIBNAME statement used with the SASEDOC engine provides read access to an output object. You cannot write an output object to a library with the SASEDOC engine. However, you can delete or rename a data set.

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

Arguments

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

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

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

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

DOC_SEQNO=sequence-number

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

sequence-number

is a number which, when combined with a path name, uniquely identifies the entry in the director.

See also: the DOCUMENT procedure in SAS Output Delivery System: User's Guide

Additional LIBNAME Statement Arguments and Options

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

Details

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

Examples

Example 1: Assigning a LIBNAME to an ODS DOCUMENT

LIBNAME statement

Option:

    DOC_SEQNO=

ODS DOCUMENT statement

Option:

    NAME=

Other SAS features:

    PROC DATASETS
    PROC GLM
    PROC PRINT

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

    Overall ANOVA
    Fit statistics
    Type I model ANOVA
    Type III model ANOVA
Program

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

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

The *plant_stats* data set contains the statistical information that PROC GLM uses to create the output objects.

```sas
data plant_stats;
do month = 1 to 12;
age  = 2 + 0.3*rannor(345467);
age2 = 3 + 0.3*rannor(345467);
age3 = 4 + 0.4*rannor(345467);
output;
end;
run;
```

Create the output objects. The GLM procedure creates the output objects. For information about viewing a record of each output object that is created, see the ODS TRACE statement in *SAS Output Delivery System: User's Guide*.

```sas
proc glm;
   class month;
   model age age2 age3=month / nouni;
   manova h=month /printe;
run;
```

The *plants* data set contains the statistical information that PROC GLM uses to create the output objects.

```sas
data plants;
   input type $ @;
do block=1 to 3;
      input stemleng @;
      output;
   end;
datalines;
clarion  32.7 32.3 31.5
clinton  32.1 29.7 29.1
knox     35.7 35.9 33.1
o’neill  36.0 34.2 31.2
compost  31.8 28.0 29.2
wabash   38.2 37.8 31.9
```

webster  32.5 31.1 29.7
;
run;

Create the output objects. The GLM procedure creates the output objects. For information about viewing a record of each output object that is created, see the ODS TRACE statement in SAS Output Delivery System: User's Guide.

```
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;
```

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

```
ods document close;
```

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

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

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

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

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

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

Output

Display 5.1 Explorer Window

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

![Explorer Window](image1)

Display 5.2 The Contents of Mylib

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

![Contents of Mylib](image2)

See Also

Procedures:

Statements:

---

**PUT Statement for ODS**

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

**Valid:** in a DATA step

**Category:** File-handling

**Type:** Executable

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

**PUT** `<specification>_<ODS>_@@|@>`;

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

### Options

#### specification

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

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

- **ods-pointer-control**
  - Moves the pointer in the buffer to a specified line or column.
  - **See also:** “When the Pointer Moves Past the End of a Line” on page 83

- **variable**
  - Identifies the variable to write.

- **Featured in:** “Example 4: Creating and Using a User-Defined Table Definition Template” in *SAS Output Delivery System: User’s Guide*

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

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

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

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

  - **Tip:** By default, the order of the columns in the data component matches the order of the columns in the buffer. However, if you have specified a table definition, it might override this order. For more information, see the discussion of the ORDER_DATA=} on page 419 in the TEMPLATE procedure section.

  - **@ | @@**

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

    - **Default:** If you do not use @ or @@, then each PUT statement in a DATA step writes a new line to the buffer.
Main discussion:  “When the Pointer Moves Past the End of a Line” on page 83

Details

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

ODS column pointer controls have the following general forms:

@ods-column

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

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

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

<table>
<thead>
<tr>
<th>Variable or expression</th>
<th>SAS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>not an integer</td>
<td>truncates the decimal portion and uses only the integer value</td>
</tr>
<tr>
<td>0 or negative</td>
<td>moves the pointer to column 1</td>
</tr>
</tbody>
</table>

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

1  writes the current line

2  moves the pointer to the first ODS column on the next line

3  continues to process the PUT statement.

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


+ods-column

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

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

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

<table>
<thead>
<tr>
<th>ods-column</th>
<th>SAS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>a positive integer</td>
<td>moves the pointer to the right</td>
</tr>
<tr>
<td>a negative integer</td>
<td>moves the pointer to the left</td>
</tr>
<tr>
<td>0</td>
<td>pointer does not move</td>
</tr>
</tbody>
</table>
Tip: If the current column position becomes less than 1, then the pointer moves to column 1. If the current column position exceeds the number of columns in the data component, then ODS:

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

See also: “Example 4: Creating and Using a User-Defined Table Definition Template” in SAS Output Delivery System: User’s Guide

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

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

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

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

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

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


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

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

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

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

“FILE Statement for ODS” on page 68

ODS _ALL_ CLOSE Statement

Closes all open ODS output destinations

Valid: anywhere
Category: ODS: Output Control

Syntax

ODS _ALL_ CLOSE;

Details

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

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

ODS CHTML Statement

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

Valid: anywhere
Category: ODS: Third-Party Formatted

Syntax

ODS CHTML <(<ID=>identifier)> <action>);
ODS CHTML <(<ID=>identifier)> <file-specification(s)> <option(s)>;

Options

The ODS CHTML statement is part of the MARKUP statement family. For a complete list of options, see the “ODS MARKUP Statement” on page 109.
Details

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

ODS CSVALL Statement

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

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS CSVALL < (<ID=>identifier)> <action>;
ODS CSVALL <(<ID=>identifier)> <file-specification(s)><option(s)>;

Options

For a complete list of options, see the “ODS MARKUP Statement” on page 109.

Details

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

ODS DECIMAL_ALIGN Statement

Controls the justification of numeric columns when no justification is specified

Valid: anywhere

Category: ODS: SAS Formatted

See: “How Are Values in Table Columns Justified?” on page 512

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

Default: ODS NO_DECIMAL_ALIGN
Chapter 5

Syntax

ODS DECIMAL_ALIGN | NO_DECIMAL_ALIGN;

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

Alias: ODS DECIMAL_ALIGN=YES

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

Alias: ODS DECIMAL_ALIGN=NO

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

ODS DOCBOOK Statement

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

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS DOCBOOK <(<ID=>identifier)> <action>);
ODS DOCBOOK <(<ID=>identifier)> <file-specification(s)> <option(s)>);

Options

For a complete list of options, see the “ODS MARKUP Statement” on page 109.

Details

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

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

Valid: anywhere
Category: ODS: Output Control

Syntax

ODS DOCUMENT action;

ODS DOCUMENT <NAME=<libname.>memname
(access-option)> <DIR=(<PATH=path>(access-option) <LABEL="label">))>
<CATALOG=permanent-catalog | _NULL_;

Actions

An action does one of the following:
- closes the destination.
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

An action can be any one of the following:

CLOSE
- closes the destination and any files that are associated with it.
  Tip: When an ODS destination is closed, ODS does not send output to that destination. Closing an unneeded destination frees some system resources.

EXCLUDE exclusion(s) | ALL | NONE
- excludes one or more output objects from the DOCUMENT destination.
  Default: NONE
  Restriction: The DOCUMENT destination must be open for this action to take effect.
  Main discussion: “ODS EXCLUDE Statement” on page 90

SELECT selection(s) | ALL | NONE
- selects one or more output objects for the DOCUMENT destination.
  Default: ALL
  Restriction: The DOCUMENT destination must be open for this action to take effect.
  Main discussion: “ODS SELECT Statement” on page 188

SHOW
- writes the current selection or exclusion list for the destination to the SAS log.
  Restriction: The destination must be open for this action to take effect.
  Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.
  See also: “ODS SHOW Statement” on page 197
Options

CATALOG=permanent-catalog | _NULL_

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

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

The *permanent catalog* has the following form:

```
<libname.><memname>;
```

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

**Alias:** CAT=

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

DIR=

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

**LABEL=label**

assigns a label to a path.

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

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

**PATH=**

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

*path*
can have the form:

```
path<#sequence-number>
```

where

*path*
is the name of the path.

*#sequence-number*
is a number which, when combined with a path name, uniquely identifies the entry in the directory that contains it.

**Default:** The default path is “\” (root).
Tip: You can specify a directory that contains entries that do not exist in the document.

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

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

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

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

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

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

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

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

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

Default: UPDATE

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

**NAME=**

<libname.>memname<(access-option)>

**libname**
specifies the SAS library where the document is stored.

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

**memname**
specifies the document name.

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

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

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

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

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

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

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

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

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

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

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

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

---

**ODS EXCLUDE Statement**

Specifies output objects to exclude from ODS destinations

- **Valid:** anywhere
- **Category:** ODS: Output Control

**Syntax**

```sas
ODS <ODS-destination> EXCLUDE exclusion(s) | ALL | NONE;
```

**Arguments**

- **exclusion(s)** specifies one or more output objects to add to an exclusion list.
  
  By default, ODS automatically modifies exclusion lists at the end of a DATA step that uses ODS, or at the end of a procedure step. For information about modifying these lists, see “Selection and Exclusion Lists” on page 34.

  For information ending a procedure or DATA step, see the section on DATA Step Processing in *SAS Language Reference: Concepts*.

- Each exclusion has the following form:

  ```sas
  output-object <(PERSIST)>
  ```

- **output-object** specifies one or more output objects to exclude. To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. You can specify an output object as

  - a full path. For example,
    ```sas
    Univariate.City_Pop_90.TestsForLocation
    ```
    
    is the full path of the output object.

  - a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is
    ```sas
    Univariate.City_Pop_90.TestsForLocation
    ```
    then the partial paths are:
    ```sas
    City_Pop_90.TestsForLocation
    TestsForLocation
    ```
a label that is enclosed by quotation marks.
For example,

"The UNIVARIATE Procedure"

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

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

Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement.

a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is

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

then the partial label paths are:

"CityPop_90"."Tests For Location"
"Tests For Location"

a mixture of labels and paths.

any of the partial path specifications, followed by a pound sign (#) and a number. For example, TestsForLocation#3 refers to the third output object that is named TestsForLocation.

See also: “ODS TRACE Statement” on page 197.

(PERSIST)
keeps the output-object that precedes the PERSIST option in the exclusion list, even if the DATA or procedure step ends, until you explicitly modify the list with

- 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.

Requirement: You must enclose PERSIST in parentheses.

ALL
specifies that ODS does not send any output objects to the open destination.

Alias: ODS EXCLUDE DEFAULT

Interaction: If you specify ALL without specifying a destination, ODS sets the overall list to EXCLUDE ALL and sets all other lists to their defaults.

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

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

NONE
specifies that ODS send all of the output objects to the open destination.

Interaction: If you specify the NONE argument without specifying a destination, ODS sets the overall list to EXCLUDE NONE and sets all other lists to their defaults.
Tip: ODS EXCLUDE NONE has the same effect as ODS SELECT ALL.
Tip: To temporarily suspend a destination, use ODS SELECT NONE. Use ODS SELECT ALL when you want to resume sending output to the suspended destination.

Options

**ODS-destination**
specifies which ODS destination’s exclusion list to write to, where *ODS-destination* can be any valid ODS destination. For a discussion of ODS destinations, see “What Are the ODS Destinations?” on page 25.

**Default:** If you omit *ODS-destination*, ODS writes to the overall exclusion list.

**Tip:** To set the exclusion list for the Output destination to something other than the default, use the “ODS OUTPUT Statement” on page 135.

Details

Although you can maintain a selection list for one destination and an exclusion list for another, the results are less complicated if you maintain the same types of lists for all the destinations that you route output to.

See Also

Statements:
- “ODS SELECT Statement” on page 188
- “ODS SHOW Statement” on page 197
- “ODS TRACE Statement” on page 197

**ODS GRAPHICS Statement (Experimental)**

Enables ODS automatic graphic capabilities

**CAUTION:**
The ODS GRAPHICS statement is an experimental feature that is available in SAS 9.1. Do not use the ODS GRAPHICS statement in production jobs.

**Valid:** anywhere

**Category:** ODS: Output Control

**Default:** OFF

**Restriction:** ODS statistical graphics do not create output for the LISTING destination. You must send your output to at least one other ODS destination (such as HTML, RTF, DOCUMENT, Printer family) to obtain output.

**Restriction:** ODS statistical graphics do not support any SAS/GRAPH global statements (such as GOPTIONS, SYMBOL, PATTERN).

**Restriction:** ODS statistical graphics do not support the GTITLE or GFOOTNOTE options available with the ODS destinations HTML, RTF, and MARKUP.

**Restriction:** ODS statistical graphics do not support the ODS USEGOPT statement.
See also: For more information about ODS statistical graphics and the procedures that produce them, see the section about statistical graphics using ODS in SAS/STAT User’s Guide.

Syntax

ODS GRAPHICS < OFF | ON </options>>;

Arguments

OFF
turns off the automatic ODS graphic generation.

ON
turns on the automatic ODS graphic generation.

Options

ANTIALIAS = OFF | ON
controls the smoothing of the components in a graph. All text displayed on the graph will always be anti-aliased.

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

Alias: NOANTIALIAS

ON
smoothes jagged edges of all components in the graph.

Alias: ANTIALIAS

Restriction: If the number of observations in the data set exceeds 250, then ANTIALIAS= is turned off, even if you specify the option ANTIALIAS=ON.

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

Note: This feature only effects the ODS statistical graphics features, and has no effect on standard graphics features that already rely on the GOPTIONS values.

image-file-type
specifies the type of image you want to add to your graph. For a list of image file types and their descriptions, see “Supported Image File Types for Output Destinations” on page 94.

STATIC
dynamically uses the best quality static image format for the active output destination.

STATICMAP
dynamically uses the best quality image map format for the active output destination, and provides a map file for tool tips.

Restriction: If the number of observations in the data set exceeds 500, then the map file is not generated.
**Default:** STATIC

`IMAGENAME= filename` specifies the base image filename. By default, the name of the output object will be used. You can determine the name of the output object by using the ODS TRACE statement. For more information, see “ODS TRACE Statement” on page 197.

**Restriction:** The base image name should not contain extension information. ODS automatically adds the increment value and the appropriate extension (which is specific to the output destination that has been selected).

`PERSIST | PERSIST=` determines when ODS clears the data cache that it is created when the ODS graphics feature is enabled.

- **PERSIST**
  clears the data cache on every RUN boundary.

- **PERSIST=PROC | RUN**
  maintains the data cache across either procedure boundaries or RUN boundaries.

  - **PROC**
    maintains the data cache across procedure boundaries.

  - **RUN**
    maintains the data cache across RUN boundaries.

**Restriction:** This value only yields different results when an interactive PROC is active; otherwise the DEFAULT/RUN settings are equivalent.

**Default:** PERSIST

`RESET` resets the index counter that is appended to static image files.

**Details**

**Supported Image File Types for Output Destinations** The following table lists all of the supported image file types for ODS output destinations.

<table>
<thead>
<tr>
<th>Output Destination</th>
<th>Supported Image File Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>GIF (default), JPEG, PNG</td>
</tr>
<tr>
<td>LATEX</td>
<td>PS (default), EPSI, GIF, JPEG, PNG</td>
</tr>
<tr>
<td>Printer Family</td>
<td>Contained in PostScript file</td>
</tr>
<tr>
<td>RTF</td>
<td>Contained in RTF file</td>
</tr>
</tbody>
</table>
Description of Supported Image File Types

<table>
<thead>
<tr>
<th>Image File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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>
</tbody>
</table>

ODS HTML Statement

Opens, manages, or closes the HTML destination, which produces HTML 4.0 output that contains embedded styleheets

Valid: anywhere

Category: ODS: Third-Party Formatted

CAUTION:

After SAS 9, the ODS HTML statement produces HTML 4.0 output that differs considerably from the HTML 3.2 output that is produced by previous versions of SAS. If you want HTML 3.2 formatting, use the ODS HTML3 statement or change the setting of the HTML version in the SAS registry. See “Changing Your Default HTML Version Setting” on page 32 for more information.

Restriction: When you open the destination, a stylesheet is written and linked to the body file. Therefore, you cannot make stylesheet changes from within your SAS program. For example, after the destination is open, changing the value of the STYLE= option has no effect. You can make style changes in either of the following ways:

- Close the destination, edit or create a new stylesheet, then submit the program again specifying the new or modified stylesheet.
- Edit the body file, changing the stylesheet url to the desired stylesheet.

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly.
because without SAS, the SAS Monospace font is not recognized. To make your
document display correctly, include the following statement before your SAS program:

```sas
OPTIONS FORMCHAR="|----|+|---+=-|\<>*";
```

### Syntax

**ODS HTML** `<(<ID=>identifier)> <action>;

**ODS HTML** `<(<ID=>identifier)> <html-file-specification(s) ><option(s)>;

### Options

For a complete list of the options, see the “ODS MARKUP Statement” on page 109.

### Details

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

### Examples

#### Example 1: Creating a Separate Body File for Each Page of Output

**ODS features:**

**ODS HTML statement:**

- **Action:**
  - CLOSE

- **Arguments:**
  - CONTENTS=
  - BODY=
  - FRAME=
  - PAGE=

- **Options:**
  - BASE=
  - NEWFILE=

**Other SAS features:**

- #BYVAL parameter in titles
- NOBYLINE|BYLINE system option
- OPTIONS statement
- PROC FORMAT
- PROC SORT
- PROC REPORT
PROC TABULATE
TITLE statement

Program Description  The following example creates a separate HTML file for each page of procedure output, as well as a table of contents, a table of pages, and a frame file. The table of contents and table of pages do not appear any different or behave any differently from those that would be created if all the output were in a single file. Because the output is in separate files, you cannot scroll from one page of output to the next. However, you can select individual HTML files to include in a report.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program

Create the input data set. The data set GRAIN_PRODUCTION contains data on the amounts of wheat, rice, and corn that the five leading grain-producing nations produced during 1995 and 1996.

```plaintext
data grain_production;
  length Country $ 3 Type $ 5;
  input Year country $ type $ Kilotons;
datalines;
1995 BRZ Wheat 1516
1995 BRZ Rice 11236
1995 BRZ Corn 36276
1995 CHN Wheat 102207
1995 CHN Rice 185226
1995 CHN Corn 112331
1995 IND Wheat 63007
1995 IND Rice 122372
1995 IND Corn 9800
1995 INS Wheat .
1995 INS Rice 49860
1995 INS Corn 8223
1995 USA Wheat 59494
1995 USA Rice 7888
1995 USA Corn 187300
1996 BRZ Wheat 3302
1996 BRZ Rice 10035
1996 BRZ Corn 31975
1996 CHN Wheat 109000
1996 CHN Rice 190100
1996 CHN Corn 119350
1996 IND Wheat 62620
1996 IND Rice 120012
1996 IND Corn 8660
1996 INS Wheat .
1996 INS Rice 51165
1996 INS Corn 8925
1996 USA Wheat 62099
1996 USA Rice 7771
```
Sort the data set. PROC SORT sorts the data, first by values of the variable **Year**, then by values of the variable **Country**, and finally by values of the variable **Type**.

```plaintext
proc sort data=grain_production;
  by year country type;
run;
```

Create a user-defined format. PROC FORMAT creates the user-defined format `$CNTRY`.

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

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

```plaintext
ods listing close;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output.

The **FRAME=**, **CONTENTS=**, and **PAGE=** options create a frame that includes a table of contents and a table of pages that link to the contents of the body file. The body file also appears in the frame. **BASE=** specifies a string to use as the first part of all links and references to the HTML files. Because no URL is specified for individual files, the final part of the link will match the filename.

**CAUTION:** The string that you specify must be a valid path to your HTML files.

```plaintext
ods html body='grain-body.htm'
  contents='grain-contents.htm'
  frame='grain-frame.htm'
  page='grain-page.htm'
  base='http://www.yourcompany.com/local-address/'
```

Specify that SAS create a new body file for each page of output. The **NEWFILE=PAGE** option opens and creates a new body file for each page of output.

```plaintext
newfile=page;
```

Suppress the default BY line and specify a new value into the BY line. The **NOBYLINE** option suppresses the default BY line variable. The **#BYVAL** parameter specification inserts the current value of the BY variable **Year** into the title.

```plaintext
options nobyline;
  title 'Leading Grain-Producing Countries';
  title2 'for #byval(year)';
```
**Produce a report.** This PROC REPORT step produces a report on grain production. Each BY group produces a page of output, so ODS creates a new body file for each BY group. The NOWINDOWS option specifies that PROC REPORT runs without the REPORT window and sends its output to the open output destination(s).

```
proc report data=grain_production nowindows;
   by year;
   column country type kilotons;
   define country / group width=14 format=$cntry.;
   define type / group 'Type of Grain';
   define kilotons / format=comma12.;
   footnote 'Measurements are in metric tons.';
run;
```

**Restore the default BY line and clear the second TITLE statement.** The BYLINE option restores the default BY line. The TITLE2 statement clears the second TITLE statement.

```
options byline;
title2;
```

**Produce a report.** The TABLE statement in this PROC TABULATE step has the variable **Year** as the page dimension. Therefore, PROC TABULATE explicitly produces one page of output for 1995 and one for 1996. ODS starts a new body file for each page.

```
proc tabulate data=grain_production format=comma12.;
   class year country type;
   var kilotons;
   table year,
       country*type,
       kilotons*sum=' ' / box=_page_ misstext='No data';
   format country $cntry.;
   footnote 'Measurements are in metric tons.';
run;
```

**Close the HTML destination.** The ODS HTML CLOSE statement closes the HTML destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files in a browser window.

```
ods html close;
```
### HTML Output

#### Display 5.3  HTML Frame File

This frame file shows the first body file. Links in the table of contents and the table of pages point to the other body files.

#### Links That Are Created in the HTML Output

These HREF= attributes from the links in the contents file point to the HTML tables that ODS creates from the PROC REPORT and PROC TABULATE steps.

- \[ \text{HREF='http://www.yourcompany.com/local-address/grain-body.htm#IDX'} \]
- \[ \text{HREF='http://www.yourcompany.com/local-address/grain-body1.htm#IDX1'} \]
- \[ \text{HREF='http://www.yourcompany.com/local-address/grain-body2.htm#IDX2'} \]
- \[ \text{HREF='http://www.yourcompany.com/local-address/grain-body3.htm#IDX3'} \]

Notice how these HREF attributes are constructed:

- The value of the BASE= option provides the first part of the HREF, which is http://www.yourcompany.com/local-address/. This part of the HREF is the same for all the links that ODS creates.

- The value of the BODY= option, \textit{grain-body}, provides the basis for the next part of the HREF. However, because the NEWFILE= option creates a new file for each output object, ODS increments this base value each time that it creates a file. The resulting file names become part of the HREF. They are \textit{grain-body.htm}, \textit{grain-body1.htm}, \textit{grain-body2.htm}, and \textit{grain-body3.htm}.

- The value of the ANCHOR= option provides the basis for the last part of the HREF, which follows the pound sign (#). Because the ANCHOR= option is not used in this example, ODS uses the default value of IDX. With each use, ODS increments the value of the anchor.
Example 2: Appending to HTML Files

ODS features:
- ODS HTML statement:
  - Argument:
    - BODY= with a fileref
    - NO_BOTTOM_MATTER suboption
    - NO_TOP_MATTER suboption
  - Options:
    - ANCHOR=
    - STYLE=
- Other SAS features:
  - FILENAME statement
  - PROC PRINT
  - PROC REPORT
  - DATA _NULL_ statement

Data set:
- GRAIN_PRODUCTION on page 97

Format:
- $CNTRY. on page 98

Program Description
The following example creates HTML output from PROC PRINT and PROC REPORT. It also uses the DATA step to write customized HTML code to the file that contains the HTML output. The DATA step executes between procedure steps.

Program

Close the LISTING destination so that no listing output is produced. The ODS LISTING statement closes the LISTING destination to conserve resources. If the destination is left open, then ODS will produce both Listing and HTML output.

```
ods listing close;
```

Assign a fileref to the file GrainReport.html. The FILENAME statement assigns the fileref REPORTS to the file GrainReport.html that will contain the HTML output.

```
filename reports ’GrainReport.html’;
```

Create HTML output and suppress the writing of the default HTML code that would be written at the end of the file. The ODS HTML statement opens the HTML destination and creates HTML output. The NO_BOTTOM_MATTER option suppresses the writing of the default HTML code that, by default, ODS writes at the end of a file.

```
ods html body=reports (no_bottom_matter)
```

Specify the style definition for formatting the HTML output. The STYLE= option specifies that the style D3D be used.

```
style=D3D;
```
Create a report that contains only the data from 1996. Select and format the variables that you want to include, specify a title, and specify a footnote. This PROC PRINT step prints the observations in the data set GRAIN_PRODUCTION that have a value of 1996 for the variable `year`. The VAR statement selects `country`, `type`, and `kilotons` as the variables that you want to be displayed in the output. The TITLE and FOOTNOTE statements specify the title and footnote.

```plaintext
proc print data=grain_production;
  var country type kilotons;
  format country $cntry. kilotons comma12.;
  where year=1996;
  title 'Leading Grain-Producing Countries';
  footnote 'Measurements are in metric tons.';
run;
```

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

```plaintext
ods html close;
```

**Assign the fileref REPORTS to the file 'GrainReport.html'.** This FILENAME statement assigns a fileref to the file to be updated, `GrainReport.html`. The MOD option opens the file in update mode.

**Operating Environment Information:** The MOD option might not be valid in all operating environments. See your operating environment documentation for more information.

```plaintext
filename reports 'GrainReport.html' mod;
```

**Append text to the HTML file REPORTS.** This DATA step writes to the file that is referenced by REPORTS. The PUT statements create an H1 header in the HTML file.

```plaintext
data _null_
  file reports;
  put '<h1>The preceding output is from PROC PRINT.';
  put 'I am going to try a variety of procedures.';
  put 'Let me know which procedure you prefer.';
  put 'By the way, this report uses the D3D style.</h1>';
run;
```

**Create HTML output.** This ODS HTML statement opens the HTML destination and creates HTML output. The NO_TOP_MATTER and the NO_BOTTOM_MATTER suboptions suppress the default HTML code that ODS writes to the top and the bottom of a file.

```plaintext
ods html body=reports (no_top_matter no_bottom_matter)
```

**Specify the root name for the HTML anchor tags.** The ANCHOR= option specifies `report` as the root name for the HTML anchor tags.

**Note:** When you use ODS to append to an HTML file that ODS created, you must specify a new anchor name each time that you open the file from ODS so that you do not write the same anchors to the file again. (ODS cannot recognize anchors that are already in the file when it opens it, and by default it uses `IDX` as the base for anchor names).

```plaintext
anchor='report';
```
**Create a report that contains only the 1996 data.** The PROC REPORT step prints the data set. ODS adds HTML output to the body file. The NOWINDOWS option specifies that PROC REPORT runs without the REPORT window and sends its output to the open output destination(s).

```plaintext
proc report data=grain_production nowindows;
   where year=1996;
   column country type kilotons;
   define country / group width=14 format=$cntry. ;
   define type / group 'Type of Grain';
   define kilotons / format=comma12. ;
run;
```

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

```plaintext
ods html close;
```

**Append text to the HTML file REPORTS.** This DATA step writes to the file that is referenced by REPORTS. The PUT statements create an H1 header in the HTML file.

```plaintext
data _null_;  
   file reports;  
   put '<h1>The preceding output is from PROC REPORT.';  
   put 'It doesn\'t repeat the name of the country on every line.';  
   put 'This report uses the default style.</h1>';  
run;
```

**Create HTML output to write the bottom matter to the file, repress the printing of the top matter, and provide a new root name for the anchor tags.** In order to write the bottom matter to the HTML file so that it contains valid HTML code, you must open the HTML destination one more time. NO_TOP_MATTER ensures that the top matter is not placed in the file again. ANCHOR= provides a new root name for the anchors in the bottom matter.

```plaintext
ods html body=reports(no_top_matter)anchor='end';
```

**Close the HTML destination.** The ODS HTML CLOSE statement closes the HTML destination and all the files that are open for that destination.

```plaintext
ods html close;
```
HTML Output

Display 5.4  HTML Output with Appended HTML

This output is created by appending HTML output to an existing HTML file.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Country</th>
<th>Type</th>
<th>Kiloliters</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Brazil</td>
<td>Wheat</td>
<td>3,902</td>
</tr>
<tr>
<td>11</td>
<td>Brazil</td>
<td>Rice</td>
<td>16,037</td>
</tr>
<tr>
<td>12</td>
<td>Brazil</td>
<td>Corn</td>
<td>51,975</td>
</tr>
<tr>
<td>13</td>
<td>China</td>
<td>Wheat</td>
<td>118,915</td>
</tr>
<tr>
<td>14</td>
<td>China</td>
<td>Rice</td>
<td>190,016</td>
</tr>
<tr>
<td>15</td>
<td>China</td>
<td>Corn</td>
<td>118,566</td>
</tr>
<tr>
<td>16</td>
<td>India</td>
<td>Wheat</td>
<td>62,820</td>
</tr>
<tr>
<td>17</td>
<td>India</td>
<td>Rice</td>
<td>120,712</td>
</tr>
<tr>
<td>18</td>
<td>India</td>
<td>Corn</td>
<td>8,066</td>
</tr>
<tr>
<td>19</td>
<td>Indonesia</td>
<td>Wheat</td>
<td>51,965</td>
</tr>
<tr>
<td>20</td>
<td>Indonesia</td>
<td>Rice</td>
<td>8,825</td>
</tr>
<tr>
<td>21</td>
<td>United States</td>
<td>Wheat</td>
<td>62,066</td>
</tr>
<tr>
<td>22</td>
<td>United States</td>
<td>Rice</td>
<td>7,771</td>
</tr>
<tr>
<td>23</td>
<td>United States</td>
<td>Corn</td>
<td>226,964</td>
</tr>
</tbody>
</table>

Measurements are in metric tons.

The preceding output is from PROC PRINT. I am going to try a variety of procedures. Let me know which procedure you prefer. By the way, this report uses the D3D style.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Country</th>
<th>Type</th>
<th>Kiloliters</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Brazil</td>
<td>Corn</td>
<td>31,975</td>
</tr>
<tr>
<td>11</td>
<td>Brazil</td>
<td>Rice</td>
<td>10,036</td>
</tr>
<tr>
<td>12</td>
<td>Brazil</td>
<td>Wheat</td>
<td>3,302</td>
</tr>
<tr>
<td>13</td>
<td>China</td>
<td>Corn</td>
<td>119,305</td>
</tr>
<tr>
<td>14</td>
<td>China</td>
<td>Rice</td>
<td>190,119</td>
</tr>
<tr>
<td>15</td>
<td>China</td>
<td>Wheat</td>
<td>103,030</td>
</tr>
<tr>
<td>16</td>
<td>India</td>
<td>Corn</td>
<td>8,850</td>
</tr>
<tr>
<td>17</td>
<td>India</td>
<td>Rice</td>
<td>120,011</td>
</tr>
<tr>
<td>18</td>
<td>Indonesia</td>
<td>Wheat</td>
<td>62,020</td>
</tr>
<tr>
<td>19</td>
<td>Indonesia</td>
<td>Rice</td>
<td>8,725</td>
</tr>
<tr>
<td>20</td>
<td>Indonesia</td>
<td>Corn</td>
<td>51,110</td>
</tr>
<tr>
<td>21</td>
<td>United States</td>
<td>Wheat</td>
<td>226,964</td>
</tr>
<tr>
<td>22</td>
<td>United States</td>
<td>Rice</td>
<td>7,771</td>
</tr>
<tr>
<td>23</td>
<td>United States</td>
<td>Corn</td>
<td>8,066</td>
</tr>
</tbody>
</table>

Measurements are in metric tons.

The preceding output is from PROC REPORT. It doesn’t repeat the name of the country on every line. This report uses the default style.
See Also

Statements:
Appendix 2, “ODS and the HTML Destination,” on page 637
“ODS MARKUP Statement” on page 109

ODS HTMLCSS Statement

Opens, manages, or closes the HTMLCSS destination, which produces HTML output with cascading style sheets
Valid: anywhere
Category: ODS: Third-Party Formatted

Syntax

ODS HTMLCSS (<ID=><identifier>) <action>;
ODS HTMLCSS (<ID=><identifier>) <file-specification(s)><option(s)>;

Options

The ODS HTMLCSS statement is part of the MARKUP statement family. For a complete list of options, see the “ODS MARKUP Statement” on page 109.

Details

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

ODS HTML3 Statement

Opens, manages, or closes the HTML3 destination, which produces HTML 3.2 formatted output
Valid: anywhere
Category: ODS: Third-Party Formatted

Syntax

ODS HTML3 (<ID=><identifier>) <action>;
ODS HTML3 (<ID=><identifier>) <html-file-specification(s)><option(s)>;
Options

For a complete list of options, see the “ODS MARKUP Statement” on page 109.

Details

The ODS HTML3 statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

By default, the SAS registry is configured to generate HTML 4 output when you specify the ODS HTML statement. To permanently change the default HTML version to 3.2, you can change the setting of the HTML version in the SAS registry. The ODS HTML statement will then produce HTML 3.2 output. For information about how to change your default HTML version, see “Changing Your Default HTML Version Setting” on page 32.

See Also

Statements:
“ODS MARKUP Statement” on page 109
“ODS HTML Statement” on page 95
Appendix 2, “ODS and the HTML Destination,” on page 637
“Changing SAS Registry Settings for ODS” on page 31
different markup languages such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.

**ODS LISTING Statement**

*Opens, manages, or closes the LISTING destination*

**Valid:** anywhere

**Category:** ODS: SAS Formatted

**Syntax**

```plaintext
ODS LISTING <action>;  
ODS LISTING <DATAPANEL=number | DATA | PAGE > <FILE=file-specification>;
```

**Without an Action or Options**

If you use the ODS LISTING statement without an action or options, it opens the LISTING destination.

**Actions**

An action does one of the following:
- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

An action can be one of the following:

**CLOSE**

Closes the LISTING 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 LISTING destination.  
**Default:** NONE  
**Restriction:** The LISTING destination must be open for this action to take effect.  
**Main discussion:** “ODS EXCLUDE Statement” on page 90

**SELECT selection(s) | ALL | NONE**

Selects output objects for the LISTING destination.  
**Default:** ALL  
**Restriction:** The LISTING destination must be open for this action to take effect.  
**Main discussion:** “ODS SELECT Statement” on page 188
SHOW writes the current selection or exclusion list for the LISTING destination to the SAS log.

 Restriction: The LISTING destination must be open for this action to take effect.
 Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.

 See also: “ODS SHOW Statement” on page 197

Options

DATAPANEL=number | DATA | PAGE suggests how to split a table that is too wide to fit on a single page into sections of columns and rows. Each section of columns and rows is a data panel. Each data panel has column headers at the top.

 Note: In this context, a page is what the procedure uses as a page in creating the Listing output. The SAS system options LINESIZE= and PAGESIZE= generally determine the page size, although some procedures (PROC REPORT, for example) can temporarily override the values that the system options specify.

 number writes the specified number of observations in a panel, if possible. More than one panel can occur on every page if space permits.

 Range: 1 to the largest integer that the operating system supports

DATA bases the size of the panel on the way the table is stored in memory. This value provides the fastest performance. However, if the table contains many columns, the number of rows in each panel might be small.

PAGE tries to make panels that match the page size. If the table contains more columns than can fit on a page, the first page is filled with as many observations as possible for as many columns as will fit on a single line. The second page contains the same observations for the next group of columns, and so on until all rows and columns have been printed.

 This arrangement will minimize the amount of space used for column headers because most pages will contain observations for only one set of columns.

 Restriction: If the page size is greater than 200, ODS uses DATAPANEL=200.

 Default: PAGE

FILE=file-specification specifies the file to write to. file-specification can be one of the following:

 'external-file'
 is the name of an external file to write to.

fileref
 is a fileref that has been assigned to an external file. Use the FILENAME statement to assign a fileref. (For information about the FILENAME statement, see SAS Language Reference: Dictionary.)

 Default: If you do not specify a file to write to, ODS writes the output to the LISTING window.
**ODS MARKUP Statement**

**Opens, manages, or closes the MARKUP destination, which produces SAS output that is formatted using one of many different markup languages**

**Valid:** anywhere

**Category:** ODS: Third-Party Formatted

**Interaction:** The output type is determined by the TAGSET | TYPE= option, which specifies the kind of markup language that is applied to the output.

**Interaction:** By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```sas
OPTIONS FORMCHAR="|----|+|---+=|-/<>*";
```

---

**Syntax**

**ODS MARKUP**

```
ODS MARKUP (<(ID=>identifier)> <action>);
ODS MARKUP (<(ID=>identifier)> <markup-file-specification(s)> <option(s)> <TAGSET=tagset-name>);
```

**Actions**

An action does one of the following:

- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

An action can be one of the following:

**CLOSE**

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

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

**EXCLUDE exclusion(s) | ALL | NONE**

excludes one or more output objects from the destination.

**Default:** NONE

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

**Main discussion:** “ODS EXCLUDE Statement” on page 90

**SELECT selection(s) | ALL | NONE**

selects output objects for the specified destination.

**Default:** ALL
Restriction: A destination must be open for this action to take effect.

Main discussion: “ODS SELECT Statement” on page 188

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

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

See also: “ODS SHOW Statement” on page 197

Tip: If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list. For information about selection and exclusion lists, see “Selection and Exclusion Lists” on page 34.

Options

ANCHOR=’anchor-name’
specifies a unique base name for the anchor tag that identifies each output object in the current body file.

Each output object has an anchor tag for the contents, page, and frame files to reference. The links and references, which are automatically created by ODS, point to the name of an anchor. Therefore, each anchor name in a file must be unique.

anchor-name
is the base name for the anchor tag that identifies each output object in the current body file.

ODS creates unique anchor names by incrementing the name that you specify. For example, if you specify ANCHOR=’tabulate’, then ODS names the first anchor tabulate. The second anchor is named tabulate1; the third is named tabulate2, and so on.

Requirement: You must enclose anchor-name in quotation marks.

Tip: You can change anchor names as often as you want by specifying the ANCHOR= option in a Markup Family statement anywhere in your program. Once you have specified an anchor name, it remains in effect until you specify a new one.

Restriction: Each anchor name in a file must be unique.

Interaction: If you open a file to append to it, then be sure to specify a new anchor name so that you do not write the same anchors to the file again. ODS does not recognize anchors that are already in a file when it opens the file.

Tip: Specifying new anchor names at various points in your program is useful when you want other web pages to link to specific parts of your markup language output. Because you can control where the anchor name changes, you know in advance what the anchor name will be at those points.

ARCHIVE=’string’
The ARCHIVE= option is only valid for the GOPTIONS java device. The ARCHIVE= option allows you to specify which applet to use in order to view the ODS HTML output.

The string must be one that the browser can interpret. For example, if the archive file is local to the machine that you are running SAS on, you can use the FILE protocol to identify the file. If you want to point to an archive file that is on a web server, use the HTTP protocol.

Default: If you do not specify ARCHIVE= and you are using the JAVA device driver, ODS uses the value of the SAS system option APPLETOC=. This value points to the location of the Java archive files that ship with the SAS system. To find out what the value of this option is, you can either look in the Options
window in the Files folder under Environment Control, or you can submit the following procedure step:

```
proc options option=appletloc;
run;
```

There is no default if you are using the ACTIVEX device driver.

**Requirement:** The ARCHIVE attribute is a feature of Java 1.1. Therefore, if you are using the Java device driver, your browser must support this version of Java. Both Internet Explorer 4.01 and Netscape 4.05 support Java 1.1.

**Interaction:** Use ARCHIVE= in conjunction with SAS/GRAPH procedures and the DEVICE=JAVA or DEVICE=ACTIVEX option in the GOPTIONS statement.

**Tip:** Typically, this option should not be used, because the SAS server automatically determines the correct SAS/Graph applets to view the ODS HTML output. However, if you have renamed the .jar files, or have other applets with which to view the ODS HTML output, this option allows you to access these applets.

**Tip:** As stated in the CODEBASE= documentation, it is recommend that you do not put a file path in your ARCHIVE option. Instead, use the CODEBASE option to specify the file path.

**ATTRIBUTES= (attribute-pair-1 ... attribute-pair-n)**

writes the specified attributes between the tags that generate dynamic graphics output.

```
attribute-pair
```

specifies the name and value of each attribute. attribute-pair has the following form:

```
'attribute-name'='attribute-value'
```

attribute-name

is the name of the attribute.

attribute-value

is the value of the attribute.

**Requirement:** You must enclose attribute-name and attribute-value in quotation marks.

**Interaction:** Use the ATTRIBUTES= option in conjunction with SAS/GRAPH procedures and with the DEVICE=JAVA, JAVAMETA, or ACTIVEX options in the GOPTIONS statement.

**See also:** SAS/GRAPH Reference, Volumes 1 and 2 for valid attributes for the following applets:

- Graph Applet
- Map Applet
- Contour Applet
- MetaView Applet

**BASE= 'base-text'**

Specifies the text to use as the first part of all links and references that ODS creates in the output files.

```
base-text
```

is the text that ODS uses as the first part of all links and references that ODS creates in the file.
Consider this specification:

```
BASE= 'http://www.your-company.com/local-url/'
```

In this case, ODS creates links that begin with the string `http://www.your-company.com/local-url/`. The appropriate `anchor-name` completes the link.

**Requirement:** You must enclose `base-text` in quotation marks.

**CHARSET=** `character-set`

specifies the character set to be generated in the `META` declaration for the HTML output.

**See:** For information about the CHARSET option, see *SAS National Language Support (NLS): User’s Guide*.

**CODEBASE='string'**

creates a file path that is used by the GOPTIONS devices. The CODEBASE file path option has two definitions, depending on the GOPTIONS device used.

For the Java device:

The CODEBASE file path points to the directory that contains the java applets (.jar files). If a CODEBASE file path is not specified, the SAS server generates a default CODEBASE file path that is based on the install location for the applets. The install location for the applets is recorded in the SAS option APPLETLOC and can be modified by a user. It is important to note that if the default path created by the SAS install is used, the HTML output file must be viewed on the machine from which the output was generated. A typical use of CODEBASE would be to specify an HTTP:// reference to the installed applets so that the output can be viewed from any machine on the Web.

**Interaction:** If you only specify CODEBASE, the SAS server will automatically generate the correct ARCHIVE= for the HTML output. This feature allows you to put the applets in one location on the web and easily generate output that points to that location. This is regardless of the type of java graphs that are generated on the ODS HTML output page. The ARCHIVE= option can be set with or without a file path. It is recommended that you do not put paths in your ARCHIVE= option. If you set an ARCHIVE= that contains a file path, that path is used in the place of any CODEBASE specification. Instead, use the CODEBASE option to specify the file path. If you specify both CODEBASE= and ARCHIVE= options, they will be used as you specified them.

For the ActiveX device:

If a CODEBASE= file path (including a required filename) is specified for an ActiveX control, then the browser will attempt to install the control from the location (including the filename) specified in the CODEBASE= file path. This happens if the control is not already installed on the user’s machine. By default, the SAS server does not generate a CODEBASE= file path for the ActiveX control. A typical use for this functionality is for a user to put the install setup for the control on the web. All web users can then generate their HTML output with a CODEBASE that points to this location. If another user (who does not have the control installed) tries to view the output, then the user will be prompted to install the control on their machine.

**ENCODING=** `local-character-set-encoding`

overrides the encoding for input or output processing (transcodes) of external files.

**See:** For information about the ENCODING= option, see *SAS National Language Support (NLS): User’s Guide*. 
GFOOTNOTE | NOGFOOTNOTE
controls the location where footnotes are printed in the graphics output.

GFOOTNOTE
prints footnotes that are created by SAS/GRAPH, which appear inside the
graph borders.

NOGFOOTNOTE
prints footnotes that are created by ODS, which will appear outside the
graph borders.

Default:  GFOOTNOTE

Restriction:  Footnotes that are displayed by a markup language statement
support all SAS/GRAPH FOOTNOTE statement options. The font must be valid
for the browser. Options that ODS cannot handle, such as text angle
specifications, are ignored. For details about the SAS/GRAPH FOOTNOTE
statement, see SAS/GRAPH Reference, Volumes 1 and 2.

Restriction:  This option applies only to SAS programs that produce SAS/GRAPH
output files.

GPATH= file-specification <(url='Uniform-Resource-Locator' | NONE)>
specifies the location for all graphics output that is generated while the
destination is open.

file-specification
specifies the file or SAS catalog to write to. Each output object that ODS
places in the file is named automatically using the SAS/GRAPH catalog entry
name as the base name and incrementing the name as necessary. For more
information about how ODS names catalog entries and external files, see
SAS/GRAPH Reference, Volumes 1 and 2. file-specification can be one of the
following:

external-file
is the name of an external file to write to.

Requirement:  You must enclose external-file in quotation marks.

fileref
is a fileref that has been assigned to an external file. Use the
FILENAME statement to assign a fileref. For information about the
FILENAME statement, see SAS Language Reference: Dictionary.

Interaction:  If you specify a fileref in the GPATH= option, then ODS
does not use information from the GPATH= option when it constructs
links.

libref,catalog
specifies a SAS catalog to write to.

URL= 'Uniform-Resource-Locator' | NONE
provides a URL for file-specification.

Uniform-Resource-Locator
is the URL you specify. ODS uses this URL instead of the file name in
all the links and references that it creates to the file.

Requirement:  You must enclose Uniform-Resource-Locator in quotation
marks.

NONE
specifies that no information from the GPATH= option appears in the
links or references.
Tip: This option is useful for building output files that may be moved from one location to another. If the links from the contents and page files are constructed with a simple URL (one name), then they will resolve, as long as the contents, page, and body files are all in the same location.

Default: If you omit the GPATH= option, then ODS stores graphics in the location that is specified by the PATH= option. If you do not specify the PATH= option, then ODS stores the graphics in the current directory. For more information, see the PATH= option on page 119.

GTITLE | NOGTITLE
controls the location where titles are printed in the graphics output.

GTITLE
prints the title that is created by SAS/GRAPH, which will appear inside the graph borders

NOGTITLE
prints the title that is created by ODS, which will appear outside the graph borders.

Default: GTITLE

Restriction: Titles that are displayed by any markup language statement support most SAS/GRAPH TITLE statement options. The font must be valid for the browser. Options that ODS cannot handle, such as text angle specifications, are ignored. For details about the SAS/GRAPH TITLE statement, see SAS/GRAPH Reference, Volumes 1 and 2.

Restriction: This option applies only to SAS programs that produce one or more SAS/GRAPH output files.

HEADTEXT= 'markup-document-head'
specifies markup tags to place between the <HEAD> and </HEAD> tags in all the files that the destination writes to.

   markup-document-head
   is the markup tags to place between the <HEAD> and </HEAD> tags.

Tip: ODS cannot parse the markup that you supply. It should be well-formed markup that is correct in the context of the <HEAD> and </HEAD> tags.

Tip: Use the HEADTEXT= option to define programs (such as JavaScript) that you can use later in the file.

(ID= identifier)
enables you to run multiple instances of the same destination at the same time. Each instance can have different options.

identifier
specifies another instance of the destination that is already open. identifier can be numeric or a series of characters that begin with a letter or an underscore. Subsequent characters can include letters, underscores, and numeric characters.

Restriction: If identifier is numeric, it must be a positive integer.

Requirement: The ID= option must be specified immediately after the ODS MARKUP/TAGSET statement keywords.

Tip: You can omit the ID= option, and instead use a name or a number to identify the instance.

Featured in: Example 1 on page 155
**markup-file-specification**

opens a markup family destination and specifies the markup file(s) to write to. These files remain open until you do one of the following:

- close the destination with either an ODS *Markup-family-destination* CLOSE statement or ODS _ALL_ CLOSE statement.
- open the same destination with a second markup family statement. This closes the first file and opens the second file.

**markup-file-specification** has the following form:

\[
\text{file-type} = \text{file-specification} < (\text{file-specification-suboption(s)})>
\]

**file-type**

associates a type of markup file with a particular *file-specification*. *file-type* can be one of the following:

**BODY**

specifies the file that contains the primary output that is created by the ODS statement.

*Note:* For some values of TAGSET=, this output will be an HTML file, for other TAGSET= values, the output will be an XML file, and so on.

**Alias:** FILE=

**Interaction:** If you use the BODY= option in an ODS markup family statement that refers to an open ODS markup destination, the option will force ODS to close the destination and all files associated with it, and then to open a new instance of the destination. For more information see “Opening and Closing the MARKUP Destination” on page 125.

**Featured in:** All examples

**CODE**

specifies the file that contains relevant style information, such as XSL (Extensible Stylesheet Language).

**CONTENTS**

specifies the file that contains a table of contents for the output.

**FRAME**

for HTML output, specifies the file that integrates the table of contents, the page contents, and the body file. If you open the frame file, then you see a table of contents, a table of pages, or both, as well as the body file.

For XLM output, FRAME= specifies the file that contains the DTD.

**Restriction:** If you specify the FRAME= argument, then you must also specify CONTENTS=, PAGE=, or both.

**PAGE**

specifies the file that contains a description of each page of the body file, and contains links to the body file. ODS produces a new page of output whenever a procedure requests a new page.

**Interaction:** The SAS system option PAGESIZE= has no effect on pages in HTML output except when you are creating batch output. For information on the PAGESIZE= option see SAS Language Reference: Dictionary.
STYLESHEET=
places the style information for markup output into an external file, or reads stylesheet information from an existing file.

Note:  By default, if you do not specifically send the information to a separate file, then the stylesheet information is included in the specified HTML file.

file-specification
specifies the file, fileref, or SAS catalog to write to.
file-specification can be one of the following:

external-file
is the name of an external file to write to.

Requirement: You must enclose external-file in quotation marks.

fileref
is a fileref that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See: For information about the FILENAME statement, see SAS Language Reference: Dictionary.

entry.markup
specifies an entry in a SAS catalog to write to.

Interaction: If you specify an entry name, you must also specify a library and catalog. See the discussion of the PATH= argument.

(file-specification-suboption(s))
provide instructions for writing the output files.
file-specification-suboptions can be one of the following:

NO_BOTTOM_MATTER
specifies that no ending markup language source code be added to the output file or.

Alias: NOBOT

Requirement: If you append text to an external file you must use a FILENAME statement with the appropriate option for the operating environment.

Interaction: The NO_BOTTOM_MATTER suboption, in conjunction with the NO_TOP_MATTER suboption, makes it possible for you to add output to an existing file and then to put your own markup language between output objects in the file.

Interaction: When you are opening a file that ODS has previously written to, you must use the ANCHOR= option to specify a new base name for the anchors in order to avoid duplicate anchors.

Tip: If you want to leave a body file in a state that you can append to with ODS, then use NO_BOTTOM_MATTER with the file-specification in the BODY= option in any markup language statement.

See also: NO_TOP_MATTER on page 116

NO_TOP_MATTER
specifies that no beginning markup language source code be added to the top of the output file. For HTML 4.0, the NO_TOP_MATTER option removes the stylesheet.

Alias: NOTOP
Requirement: If you append text to an external file you must use a FILENAME statement with the appropriate option for the operating environment.

Interaction: The NO_TOP_MATTER suboption, in conjunction with the NO_BOTTOM_MATTER suboption, makes it possible for you to add output to an existing file and then to put your own markup language between output objects in the file.

Interaction: When you are opening a file that ODS has previously written to, you must use the ANCHOR= option to specify a new base name for the anchors in order to avoid duplicate anchors.

See also: NO_BOTTOM_MATTER on page 116 and ANCHOR= on page 110

URL= 'Uniform-Resource-Locator'
provides a URL for the file-specification. ODS uses this URL (instead of the file name) in all the links and references that it creates and that point to the file.

Tip: This option is useful for building HTML files that can be moved from one location to another. The links from the contents and page files must be constructed with a single name URL, and the contents, page, and body files must all be in the same location.

Tip: You never need to specify this suboption with the FRAME= option because ODS files do not reference the frame file.

DYNAMIC
enables you to send output directly to a web server instead of writing it to a file. This option sets the value of the HTMLCONTENTTYPE= attribute. For more information see the HTMLCONTENTTYPE= attribute on page 311.

Default: If you do not specify DYNAMIC, then ODS sets the value of HTMLCONTENTTYPE= for writing to a file.

Restriction: If you specify the DYNAMIC suboption with any file-type= file-specification in the ODS HTML statement, then you must specify it for all the file-type= file-specification in the statement.

Restriction: You must specify file-specification-suboption(s) inside parentheses, next to the file-specification in the BODY=, CONTENTS=, PAGE=, or FRAME= option.

METATEXT= 'metatext-for-document-head'
specifies HTML code to use as the <META> tag between the <HEAD> and </HEAD> tags of all the HTML files that the destination writes to.

'metatext-for-document-head'
specifies the HTML code that provides the browser with information about the document that it is loading. For example, this attribute could specify the content type and the character set to use.

Default: If you do not specify METATEXT=, then ODS writes a simple <META> tag, which includes the content-type of the document and the character set to use, to all the HTML files that it creates.

Tip: ODS cannot parse the HTML code that you supply. It should be well-formed HTML code that is correct in the context of the <HEAD> tags. If you are using METATEXT= as it is intended, then your META tag should look like this:

<br />&lt;META your-metatext-is-here&gt;

Restriction: METATEXT= cannot exceed 256 characters.
NEWFILE= starting-point
creates a new body file at the specified starting-point.

starting-point
is the location in the output where you want to create a new body file. ODS automatically names new files by incrementing the name of the body file. In the following example, ODS names the first body file REPORT.XML. Additional body files are named REPORT1.XML, REPORT2.XML, and so on.

Example:
BODY= 'REPORT.XML'
starting-point can be one of the following:

BYGROUP
starts a new file for the results of each BY group.

NONE
writes all output to the body file that is currently open.

OUTPUT
starts a new body file for each output object. For SAS/GRAPH this means that ODS creates a new file for each SAS/GRAPH output file that the program generates.

Alias: TABLE

PAGE
starts a new body file for each page of output. A page break occurs when a procedure explicitly starts a new page (not because the page size was exceeded) or when you start a new procedure.

PROC
starts a new body file each time that you start a new procedure.

Default:  NONE

Tip:  If you end the file name with a number, then ODS begins incrementing with that number. In the following example, ODS names the first body file MAY5.XML. Additional body files are named MAY6.XML, MAY7.XML, and so on.

Example:
BODY= 'MAY5.XML'

NOGFOOTNOTE
See:  GFOOTNOTE | NOGFOOTNOTE options

NOGTITLE
See:  GTITLE | NOGTITLE options

PARAMETERS= (parameter-pair-1 ... parameter-pair-n)
writes the specified parameters between the tags that generate dynamic graphics output.

parameter-pair
specifies the name and value of each parameter. parameter-pair has the following form:

'parameter-name'='parameter-value'

parameter-name
is the name of the parameter.

*parameter-value*  
is the value of the parameter.

**Requirement:** You must enclose *parameter-name* and *parameter-value* in quotation marks.

**Interaction:** Use PARAMETERS= in conjunction with SAS/GRAPH procedures and the DEVICE=JAVA, JAVAMETA, or ACTIVEX options in the GOPTIONS statement.

**See also:** SAS/GRAPH Reference, Volumes 1 and 2 for valid parameters for the following applets:
- Graph Applet
- Map Applet
- Contour Applet
- MetaView Applet

**PATH=** `file-specification (URL="Uniform-Resource-Locator" | NONE)` 
specifies the location of an external file or a SAS catalog for all markup files.

*file-specification*  
specifies the file or SAS catalog to write to.

*file-specification* can be one of the following:

- **external-file**  
is the name of an external file to write to.

  **Requirement:** You must enclose *external-file* in quotation marks.

- **fileref**  
is a fileref that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

  **Interaction:** If you use a fileref in the PATH= option, then ODS does not use information from PATH= when it constructs links.

  **See:** For information about the FILENAME statement, see SAS Language Reference: Dictionary.

- **libname.catalog**  
specifies a SAS catalog to write to.

  **See:** For information about the LIBNAME statement, see SAS Language Reference: Dictionary.

**URL=** "Uniform-Resource-Locator" | NONE 
provides a URL for the *file-specification*.

*Uniform-Resource-Locator*  
is the URL you specify. ODS uses this URL instead of the file name in all the links and references that it creates to the file.

**NONE**  
specifies that no information from the PATH= option appears in the links or references.

**Tip:** This option is useful for building output files that can be moved from one location to another. The links from the contents and page files must be constructed with a single-name URL, and the contents, page, and body files must be in the same location.
RECORD_SEPARATOR= 'alternative-separator' | NONE
specifies an alternative character or string that separates lines in the output files.
Different operating environments use different separator characters. If you do not specify a record separator, then the files are formatted for the environment where you run the SAS job. However, if you are generating files for viewing in a different operating environment that uses a different separator character, then you can specify a record separator that is appropriate for the target environment.

alternative-separator
represents one or more characters in hexadecimal or ASCII format. For example, the following option specifies a record separator for a carriage return character and a linefeed character for use with an ASCII file system:

RECORD_SEPARATOR= '0D0A'x

Requirement: You must enclose alternative-separator in quotation marks.

NONE
produces the markup language that is appropriate for the environment where you run the SAS job.

Operating Environment Information: In a mainframe environment, by default, ODS produces a binary file that contains embedded record separator characters. This binary file is not restricted by the line-length restrictions on ASCII files. However, if you view the binary files in a text editor, then the lines run together.

If you want to format the files so that you can read them with a text editor, then use RECORD_SEPARATOR= NONE. In this case, ODS writes one line of markup language at a time to the file. When you use a value of NONE, the logical record length of the file that you are writing to must be at least as long as the longest line that ODS produces. If the logical record length of the file is not long enough, then the markup language might wrap to another line at an inappropriate place.

Alias:
RECESEP=
RS=
STYLE= style-definition
specifies the style definition to use in writing the output files.

style-definition
describes how to display the presentation aspects (color, font face, font size, and so on) of your SAS output. A style definition determines the overall appearance of the documents that use it. Each style definition is composed of style elements.

Main discussion: For a complete discussion of style definitions, see Chapter 9, “TEMPLATE Procedure: Creating a Style Definition,” on page 285.

See also: For information about creating your own style definitions, see Chapter 9, “TEMPLATE Procedure: Creating a Style Definition,” on page 285.

Interaction: The STYLE= option is not valid when you are creating XML output.

Default: If you do not specify a style definition, then ODS uses the file that is specified in the SAS registry subkey

ODS ➤ DESTINATIONS ➤ MARKUP ➤ Selected Style

By default, this value specifies styles.default.
Interaction: If you specify the STYLE= option on an ODS HTML4 statement and wish to change the style definition with another ODS HTML4 statement, you must close the first statement before specifying the second statement, in order for any PROC PRINT output to use the second style definition.

TAGSET= tagset-name
specifies a keyword value for a tagset. A tagset is a template that defines how to create a markup language output type from a SAS format. Tagsets produce markup output such as Hypertext Markup Language (HTML), Extensible Markup Language (XML), and LaTeX.

An alternate form for specifying a tagset is as follows:

```
ODS directory.tagset-name  file-specification(s)<option(s)>;
ODS directory.tagset-name  action;
```

A directory can be TAGSETS, a user defined entry, or a libref. By default, the tagsets that SAS supplies are located in the directory TAGSETS, which is within the item store SASUSER.TMPLMST. For more information about user defined tagsets and item stores, see Chapter 7, “TEMPLATE Procedure: Overview,” on page 261.

Alias: TYPE=

Default: If you do not specify a TAGSET= value, then the ODS MARKUP statement defaults to XML output.

Interaction: If you use the TAGSET= option in an ODS markup family statement that refers to an open ODS markup destination, then the option will force ODS to close the destination and all files associated with it, and then to open a new instance of the destination. For more information, see “Opening and Closing the MARKUP Destination” on page 125.

Tip: SAS provides a set of tagset definitions. To get a list of the tagset names that SAS supplies, plus any tagsets that you created and stored in the SASUSER.TMPLMST template store, submit the following SAS statements:

```
proc template;
  list tagsets;
run;
```

See also: For additional information about specifying tagsets, see Chapter 11, “TEMPLATE Procedure: Creating Markup Language Tagsets,” on page 551.

The values for TAGSET= can be one of the following, which are the tagsets (templates) supplied by SAS:

- CHTML
  produces compact, minimal HTML output that does not use style information. It does produce a hierarchical table of contents.
  See: “ODS CHTML Statement” on page 84

- COLORLATEX
  produces color LaTeX, which is a document preparation system for high quality typesetting. It also generates a stylesheet. The output can be generated as PDF.
  CAUTION: COLORLATEX is an experimental tagset. Do not use this tagset in production jobs.

- CSV
  produces tabular output that contains columns of data values that are separated by commas.
**CAUTION:**
CSV is an experimental tagset. Do not use this tagset in production jobs.

**Featured in:** Creating Different Data Delimiters in a TagsetExample 8 on page 605

CSVALL
produces tabular output with titles that contain columns of data values that are separated by commas.

**See also:** “ODS CSVALL Statement” on page 85

CSVBYLINE
produces output with comma-separated values and columns of data that are separated by commas.

**CAUTION:**
CSVBYLINE is an experimental tagset. Do not use this tagset in production jobs.

XML
produces XML output.

DOCBOOK
produces XML output that conforms to the DocBook DTD by OASIS.

**See also:** “ODS DOCBOOK Statement” on page 86

EVENT_MAP
creates XML output that shows which events are being triggered and which variables are used by an event to send output from a SAS process to an output file. When you run a SAS process with EVENT_MAP, ODS writes XML to an output file that shows all event names and variable names as tags. The output helps you to create your own tagsets.

GRAPH
produces markup for graphical output that is produced by SAS/GRAPH.

HTML4
produces HTML 4.0 embedded stylesheets.

**See also:** “ODS HTML Statement” on page 95

HTMMLCSS
produces HTML output with cascading stylesheets that is similar to ODS HTML output.

**See also:** “ODS HTMMLCSS Statement” on page 105

IMODE
produces HTML output as a column of output that is separated by lines. This tagset is used by the Japanese telephone service provider, NTT.

**See also:** “ODS IMODE Statement” on page 106

LATEX
produces LaTeX, which is a document preparation system for high-quality typesetting. It also generates a stylesheet. The output can be generated as PDF.

**CAUTION:**
LATEX is an experimental tagset. Do not use this tagset in production jobs.

MSOFFICE_HTML
produces HTML code for output generated by ODS for Microsoft Office products.
MVSHTML
produces URLs within HTML files that are used in the OS/390 operating environment.

**CAUTION:**
MVSHTML is an experimental tagset. Do not use this tagset in production jobs.

NAMEDHTML
creates HTML output similar to STYLE_POPUP on page 124, but with all the objects labeled as they are when using ODS TRACE.

**CAUTION:**
NAMEDHTML is an experimental tagset. Do not use this tagset in production jobs.

ODSSTYLE
creates PROC TEMPLATE code for the STYLESHEET= option. The output helps you to create and modify style definitions.

**CAUTION:**
ODSSTYLE is an experimental tagset. Do not use this tagset in production jobs.

ODSXRPCS
produces an ODS XML remote program command stream.

**CAUTION:**
ODSXRPCS is an experimental tagset. Do not use this tagset in production jobs.

PHTML
produces simple HTML output that uses twelve style elements and no class attributes.

See also: “ODS PHTML Statement” on page 159

PYX
produces PYX, which is a simple, line-oriented notation used by Pyxie to describe the information communicated by an XML parser to an XML application. Pyxie is an Open Source library for processing XML with the Python programming language.

**CAUTION:**
PYX is an experimental tagset. Do not use this tagset in production jobs.

SASFMT
produces format markup tags that you create for the XML engine.

SASXMISS
produces alternate missing-value markup tags for the XML engine.

SASXMNSP
produces alternate “no space in text” value markup for the XML engine.

SASXMOG
produce generic XML code that is similar to the Oracle8iXML implementation used by ORACLE.

*Note:* This is the tagset used by the SAS XML LIBNAME engine for the XMLTYPE= GENERIC option.

SASXMOH
produces very simple HTML output.
Note: This is the tagset used by the SAS XML LIBNAME engine for XMLTYPE= HTML.

SASXMOIM
produces XML code that is supported by the Open Information Model (Database Schema Model) proposed by the Metadata Coalition (MDC) as vendor- and technology-independent, conforming to the XML 1.0 specification.

Note: This is the tagset used by the SAS XML LIBNAME engine for the XMLTYPE= OIMDBM option.

SASXMOR
produces XML that is equivalent to the Oracle8iXML implementation, which is used by ORACLE.

Note: This is the tagset used by the SAS XML LIBNAME engine for XMLTYPE= ORACLE.

SHORT_MAP
creates a subset of the XML output that is created by the EVENT_MAP tagset.

CAUTION:
SHORT_MAP is an experimental tagset. Do not use this tagset in production jobs.

STATGRAPH
produces markup for statistical graphs that are generated by SAS procedures.

STYLE_DISPLAY
creates a sample page of HTML output that is similar to STYLE_POPUP output. The output helps you to create and modify styles.

CAUTION:
STYLE_DISPLAY is an experimental tagset. Do not use this tagset in production jobs.

See also: STYLE_POPUP on page 124

STYLE_POPUP
creates HTML like HTMLCSS, but if you’re using Internet Explorer, STYLE_POPUP displays a window that shows the resolved ODS style definition for any item that you select.

CAUTION:
STYLE_POPUP is an experimental tagset. Do not use this tagset in production jobs.

TEXT_MAP
creates text output that shows which events are being triggered as ODS handles the output objects.

CAUTION:
TEXT_MAP is an experimental tagset. Do not use this tagset in production jobs.

Tip: You can use the TEXT_MAP output as an alternative to the output that is created by the EVENT_MAP tagset.

See also: EVENT_MAP on page 122

TPL_STYLE_LIST
creates HTML output in a bulleted list similar to EVENT_MAP but lists only a subset of the possible attributes.
CAUTION:
TPL_STYLE_LIST is an experimental tagset. Do not use this tagset in production jobs.

Tip: The output helps you to understand tagsets and styles.

TPL_STYLE_MAP
creates XML output similar to EVENT_MAP but lists only a subset of the possible attributes.

CAUTION:
TPL_STYLE_MAP is an experimental tagset. Do not use this tagset in production jobs.

Tip: The output helps you to understand tagsets and styles.

TROFF
produces Troff code, which is a text-formatting language used for high-quality photo typesetters and laser printers.

CAUTION:
TROFF is an experimental tagset. Do not use this tagset in production jobs.

user-defined-tagset
specifies the tagset that you created using PROC TEMPLATE.

Main discussion: “Creating Your Own Tagsets” on page 585

WML
uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a list of URLs as a table of contents.

See also: “ODS WML Statement” on page 206

WMLOLIST
uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with an option list for the table of contents. For more information, see Wireless Application Protocol.

CAUTION:
WMLOLIST is an experimental tagset. Do not use this tagset in production jobs.

Default: XML

TRANTAB= 'translation-table'
specifies the translation table to use when transcoding a file for output.


Details

Opening and Closing the MARKUP Destination
You can modify an open MARKUP destination with many ODS MARKUP options. However, the BODY= and TAGSET= options will automatically close the open destination that is referred to in the ODS MARKUP statement, and will also close any files associated with it, and then will open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

Specifying Multiple ODS Destinations
The ODS MARKUP statement opens or closes one destination. Like all single output destinations, you can have only one markup destination open at one time, unless you use the ID= option.
However, you can specify multiple simultaneous ODS destinations to produce multiple markup output by doing both of the following:

- specifying some of the TAGSET= value keywords as a destination
- specifying any two-level tagset name, such as TAGSETS.PYX, TAGSETS.STYLE_DISPLAY, or one of your own tagset names.

**Specifying a Tagset Keyword as an ODS Destination**

You can specify some tagset keywords as ODS destinations. The tagset determines the type of markup that you will have in your output file. For example, either of the following sets of statements are acceptable:

```plaintext
ods markup body='class.html' tagset=phtml;
...more SAS statements...
ods markup close;

ods phtml body='class.html';
...more SAS statements...
ods phtml close;
```

The ODS statement that you use to close a destination must be in the same form as the ODS statement that you used to open the destination. Therefore, the following is not acceptable, because SAS considers MARKUP and PHTML as separate destinations.

```plaintext
ods markup body='class.html' tagset=phtml;
...more SAS statements...
ods phtml close;
```

The tagsets that you can specify as both a TAGSET= value for ODS MARKUP or as a separate ODS destination are as follows:

CHTML
CSV
CSVALL
DOCBOOK
HTML4
HTMLCSS
IMODE
LATEX
PHTML
SASREPORT
TROFF
WML
WMLOLIST

**Specifying a Two-Level Tagset Name as an ODS Destination**

You can open a destination by specifying the markup that you want to produce by naming its two-level tagset name. You can specify all tagsets in this manner. For example, the following ODS statements open the SASIOXML and MYTAGSET destinations. The ODS _ALL_ CLOSE statement closes the SASIOXML and MYTAGSET destinations as well as all other open destinations.
ods tagsets.sasioxml body='test1.xml';
ods tagsets.mytagset body='test2.xml';
...more SAS statements...
ods _all_ close;

You can also specify tagset names as follows, using the TYPE= option with a two-level tagset name:

ods markup type=tagsets.sasioxml body='test.xml';

Examples

Example 1: Creating an XML File

ODS features:
ODS LISTING statement:
  Action: CLOSE
ODS MARKUP statement:
  Action: CLOSE
  Arguments:
    BODY=

Other SAS features:
PROC PRINT

Data Set:
“Creating the Statepop Data Set” on page 620

Program Description The following is an ODS MARKUP example that creates XML markup from PRINT procedure output. The TAGSET= option for the ODS MARKUP statement is not specified, which defaults to XML output.

Program

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

ods listing close;

Create XML output. The ODS MARKUP BODY= statement creates an XML file.

ods markup body='population.xml';

Print the data set. The PRINT procedure prints the data set state.pop.

proc print data=statepop;
run;
Close the MARKUP destination. The ODS MARKUP CLOSE statement closes the MARKUP destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```ods markup close;```

XML Output  The following partial output is tagged with XML (Extensible Markup Language) tags.
<?xml version="1.0" encoding="windows-1252"?>
<odsxml>
<head>
<meta operator="user"/>
</head>
<body>
<proc name="Univariate">
<label name="IDX"/>
<title class="SystemTitle" toc-level="1">US Census of Population and Housing</title>
<proc-title class="ProcTitle" toc-level="1">The UNIVARIATE Procedure</proc-title>
<proc-title class="ProcTitle" toc-level="1">Variable: CityPop_90 (1990 metropolitan pop in millions)</proc-title>
<branch name="Univariate" label="The Univariate Procedure" class="ContentProcName" toc-level="1">
<branch name="CityPop_90" label="CityPop_90" class="ContentFolder" toc-level="2">
<leaf name="Moments" label="Moments" clabel="Moments">
<output name="Moments" clabel="Moments">
<output-object type="table" class="Table">
<style>
<border spacing="1" padding="7" rules="groups" frame="box"/>
</style>
<colspecs columns="4">
<colgroup>
<colspec name="1" width="15" type="string"/>
<colspec name="2" width="10" align="right" type="string"/>
<colspec name="3" width="16" type="string"/>
<colspec name="4" width="10" align="right" type="string"/>
</colgroup>
</colspecs>
<output-head>
<row>
<header type="string" class="Header" row="1" column="1" column-end="4">
<style>
<span columns="4"/>
</style>
<value>Moments</value>
</header>
</row>
<output-body>
... more tagged output ...
</output-body>
</output>
</leaf>
</branch>
</branch>
<footnote class="SystemFooter" toc-level="1">^super *This is a ^S={foreground=black}footnote.</footnote>
</proc>
</body>
</odsxml>
Example 2: Creating an XML File and a DTD

ODS features:

ODS LISTING statement:

Action:

CLOSE

ODS MARKUP statement:

Action:

CLOSE

Arguments:

BODY=

FRAME=

TAGSET=

Other SAS features:

PROC UNIVARIATE

Data Set:

“Creating the Statepop Data Set” on page 620

Program Description

The following ODS MARKUP example creates an XML file and its Document Type Definition (DTD) related information document from PROC UNIVARIATE output.

Program

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources.

ods listing close;

Create XML output and a DTD. The ODS MARKUP BODY= statement creates an XML file. The FRAME= option specifies that you want the DTD in a frame file, and the TAGSET= option specifies that you want the default tagset, which is XML.

ods markup body='statepop.xml'
    frame='statepop.dtd' tagset=default;

Generate the statistical tables for the analysis variables. The PROC UNIVARIATE statement calculates univariate statistics for numeric variables in the STATEPOP data set. The VAR statement specifies the analysis variables and their order in the output. The TITLE statement specifies a title for the output object.

proc univariate data=statepop;
    var citypop_90 citypop_80;
    title 'US Census of Population and Housing';
run;
Close the MARKUP destination. The ODS MARKUP CLOSE statement closes the MARKUP destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```ods markup close;```

Output  This DTD specifies how the markup tags in a group of SGML or XML documents should be interpreted by an application that displays, prints, or otherwise processes the documents.

**Output 5.2**  DTD Created by the ODS MARKUP Statement

```xml
<!ELEMENT odsxml (head?,body)>  
<!ELEMENT head (meta|css)*>  
<!ELEMENT body ((label|page)*|proc)+>  
<!ELEMENT meta EMPTY>  
<!ATTLIST meta  
  operator CDATA #IMPLIED  
  author CDATA #IMPLIED>  
<!ELEMENT css EMPTY>  
<!ATTLIST css  
  file CDATA #IMPLIED>  
<!ELEMENT label EMPTY>  
<!ATTLIST label  
  name ID #IMPLIED>  
<!ELEMENT proc (title|proc-title|note|page|label|style|branch|output)*>  
<!ATTLIST proc  
  class CDATA #IMPLIED>  
... more tagged output ...
<!ELEMENT br EMPTY>  
<!ELEMENT page EMPTY>  
<!ELEMENT b (#PCDATA|it|b|ul)*>  
<!ELEMENT ul (#PCDATA|it|b|ul)*>  
<!ELEMENT it (#PCDATA|it|b|ul)*>  
<!ELEMENT style (span|align|border)*>  
<!ELEMENT span EMPTY>  
<!ATTLIST span  
  columns CDATA #IMPLIED  
  rows CDATA #IMPLIED>  
<!ELEMENT align EMPTY>  
<!ATTLIST align  
  horiz (left|center|right|justify) "left">  
<!ELEMENT border EMPTY>  
<!ATTLIST border  
  rules (none|groups|rows|cols|all) #IMPLIED  
  frame (void|above|below|hsides|lhs|rhs|vsides|box|border) #IMPLIED  
  padding CDATA #IMPLIED  
  spacing CDATA #IMPLIED>
```
Example 3: Creating Multiple Markup Output

ODS features:
- ODS LISTING statement:
  - Action: CLOSE
- ODS CSVALL statement:
  - Arguments: BODY=
- ODS MARKUP statement:
  - Arguments: BODY= TAGSET=

Other SAS features:
- OPTIONS statement
- PROC PRINT
- TITLE statement

Data set:
- GRAIN_PRODUCTION on page 97

Format:
- $CNTRY on page 98

Program Description
The following ODS example creates two different types of markup output from the same procedure output. To create two markup outputs requires two ODS destinations. Because ODS MARKUP is considered one destination, you cannot specify two tagsets without the use of the ID= option. However, you can specify one output using ODS MARKUP, then specify the other output using ODS syntax in which you specify the tagset as the destination.

Program

**Close the LISTING destination so that no listing output is produced.** The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources. The OPTIONS statement specifies that only fifteen observations be used.

```sas
ods listing close;
options obs=15;
```

**Create tabular output.** The ODS CSVALL statement produces tabular output with titles that contain columns of data values that are separated by commas

```sas
ods csvall body='procprintcsvall.csv';
```

**Create CHTML output.** The ODS MARKUP TAGSET=CHTML statement produces compact, minimal HTML output that does not use style information, and a hierarchical table of contents.

```sas
ods markup tagset=chtml body='procprintchtml.html';
```
Print the data set. The PRINT procedure prints the data set `grain_production`. The TITLE statement specifies the title.

```plaintext
title 'Leading Grain-Producing Countries';
proc print data=grain_production;
run;
```

Close the open destinations so that you can view or print the output. The ODS CSVALL CLOSE statement closes the CSVALL destination and all of the files that are associated with it. The ODS MARKUP TAGSET=CHTML L CLOSE statement closes the MARKUP destination and all of the files that are associated with it. You must close the destinations before you can view the output with a browser or before you can send the output to a physical printer.

```plaintext
ods csvall close;
ods markup tagset=chtml close;

Output

Display 5.5 CHTML Output

The following output was created by specifying the MARKUP TAGSET=CHTML statement.

### Leading Grain-Producing Countries

<table>
<thead>
<tr>
<th>Obs</th>
<th>Country</th>
<th>Type</th>
<th>Year</th>
<th>Kilograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BRZ</td>
<td>Wheat</td>
<td>1995</td>
<td>1516</td>
</tr>
<tr>
<td>2</td>
<td>BRZ</td>
<td>Rice</td>
<td>1995</td>
<td>11236</td>
</tr>
<tr>
<td>3</td>
<td>BRZ</td>
<td>Com</td>
<td>1995</td>
<td>30276</td>
</tr>
<tr>
<td>4</td>
<td>CHN</td>
<td>Wheat</td>
<td>1995</td>
<td>102207</td>
</tr>
<tr>
<td>5</td>
<td>CHN</td>
<td>Rice</td>
<td>1995</td>
<td>185226</td>
</tr>
<tr>
<td>6</td>
<td>CHN</td>
<td>Com</td>
<td>1995</td>
<td>112331</td>
</tr>
<tr>
<td>7</td>
<td>IND</td>
<td>Wheat</td>
<td>1995</td>
<td>63007</td>
</tr>
<tr>
<td>8</td>
<td>IND</td>
<td>Rice</td>
<td>1995</td>
<td>122372</td>
</tr>
<tr>
<td>9</td>
<td>IND</td>
<td>Com</td>
<td>1995</td>
<td>9880</td>
</tr>
<tr>
<td>10</td>
<td>INS</td>
<td>Wheat</td>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>INS</td>
<td>Rice</td>
<td>1995</td>
<td>49060</td>
</tr>
<tr>
<td>12</td>
<td>INS</td>
<td>Com</td>
<td>1995</td>
<td>8223</td>
</tr>
<tr>
<td>13</td>
<td>USA</td>
<td>Wheat</td>
<td>1995</td>
<td>50494</td>
</tr>
<tr>
<td>14</td>
<td>USA</td>
<td>Rice</td>
<td>1995</td>
<td>7888</td>
</tr>
<tr>
<td>15</td>
<td>USA</td>
<td>Com</td>
<td>1995</td>
<td>187390</td>
</tr>
</tbody>
</table>
Example 4: Specifying Tagset Names as ODS Destinations

When you specify tagsets and two-level tagset names as destinations, you can open and close multiple destinations, producing multiple markup output. For example:

    ods htmlcss body='test1.html';
    ods phtml body='test2.html';
    ods chtml body='test3.html';
    ods markup body='test1.xml';
    ods tagsets.event_map body='test2.xml';
    ...more SAS statements...
    ods htmlcss close;
    ...more SAS statements...
    ods chtml close;
    ...more SAS statements...
    ods _all_ close;
**ODS OUTPUT Statement**

Produces a SAS data set from an output object and manages the selection and exclusion lists for the OUTPUT destination

Valid: anywhere

Category: ODS: SAS Formatted

**Syntax**

ODS OUTPUT action;

ODS OUTPUT data-set-definition(s);

**Actions**

An action can be one of the following:

**CLEAR**
sets the list for the OUTPUT destination to EXCLUDE ALL.

**CLOSE**
closes the OUTPUT destination. When an ODS destination is closed, ODS does not send output to that destination. Closing a destination frees some system resources.

**SHOW**
writes to the SAS log the current selection or exclusion list for the OUTPUT destination. If the list is the default list (EXCLUDE ALL), then SHOW also writes the current overall selection or exclusion list.

**Arguments**

**data-set-definition**
provides instructions for turning an output object into a SAS data set. ODS maintains a list of these definitions. This list is the selection list for the OUTPUT destination. For information about how ODS manages this list, see “Selection and Exclusion Lists” on page 34. Each data-set-definition has the following form:

output-object-specification<=SAS-data-set>

where

output-object-specification
has the following form:

output-object<(MATCH_ALL<=macro-var-name> PERSIST=PROC | RUN)>

output-object identifies one or more output objects to turn into a SAS data set.

To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. For more information, about the ODS TRACE
statement see *SAS Output Delivery System: User's Guide*. You can specify an output object as any of the following:

- a full path. For example,
  
  `Univariate.City_Pop_90.TestsForLocation`

  is the full path of the output object.

- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is
  
  `Univariate.City_Pop_90.TestsForLocation`

  then the partial paths are:

  `City_Pop_90.TestsForLocation`
  `TestsForLocation`

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

  `"Tests For Location"`

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

  
  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement.

- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is
  
  `"The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"`

  then the partial label paths are:

  `"CityPop_90"."Tests For Location"`
  `"Tests For Location"`

- a mixture of labels and paths.
- any of the partial path specifications, followed by a pound sign (#) and a number. For example, `TestsForLocation#3` refers to the third output object that is named `TestsForLocation`.

**Tip:** To create multiple data sets from the same output object, list the output object as many times as you want. Each time that you list the output object, specify a different data set.

**MATCH_ALL=**<macro-var-name>

creates a new data set for each output object. For an explanation of how ODS names these data sets, see the discussion of SAS-data-set on page 137.

**macro-var-name**

specifies the macro variable where a list of all the data sets that are created are stored. Thus, if you want to concatenate all the data sets after the PROC step, then you can use the macro variable to specify all the data sets in a DATA step.

**Tip:** The MATCH_ALL option is not needed to merge conflicting output objects into one data set.
CAUTION:
A data set that is produced by SAS 9.1 without MATCH_ALL will not necessarily be identical to a data set produced by SAS 9.0 with MATCH_ALL and then concatenated in a data step. With SAS 9.0, merging dissimilar output objects with the MATCH_ALL option could result in missing columns or truncated variables. With SAS 9.1, these restrictions do not apply. For more information about merging output objects, see “Merging Dissimilar Output Objects into One Data Set” on page 137.

PERSIST=PROC | RUN
determines when ODS closes any data sets that it is creating, and determines when ODS removes output objects from the selection list for the OUTPUT destination.

PROC
maintains the list of definitions even after the procedure ends, until you explicitly modify it. To modify the list, use ODS OUTPUT with one or more data-set-specifications. To set the list for the OUTPUT destination to EXCLUDE ALL, use the following statement:

ods output clear;

RUN
maintains the list of definitions and keeps open the data sets that it is creating even if the procedure or DATA step ends, or until you explicitly modify the list.

See also: “How Does ODS Determine the Destinations for an Output Object?” on page 35

SAS-data-set
names the output data set. You can use a one-level or two-level (with a libref) name.

If you are creating a single data set, then the ODS OUTPUT statement simply uses the name that you specify. If you are creating multiple data sets with MATCH_ALL, then the ODS OUTPUT statement appends numbers to the name. For example, if you specify test as SAS-data-set and you create three data sets, then ODS names the first data set test. The additional data sets are named test1 and test2.

Note: If you end the file name with a number, then ODS begins incrementing the name of the file with that number. For example, if you specify may5 as SAS-data-set and you create three data sets, then ODS names the first data set may5. The additional data sets are named may6 and may7. △

Default: If you do not specify a data set, then ODS names the output data set DATA\text{\textit{n}}, where \textit{n} is the smallest integer that makes the name unique.

Tip: You can specify data set options in parentheses immediately after SAS-data-set.

SHOW
functions just like the ODS SHOW statement except that it writes only the selection or exclusion list for the OUTPUT destination.

Details

Merging Dissimilar Output Objects into One Data Set
By default, the ODS OUTPUT statement puts all output objects that have the same output-path into one SAS data set,
regardless of any confictions between variables in the output objects. Variables created by a later output object will get a value of missing in the observations created by the earlier output object. Variables created by an earlier output object that do not exist in a subsequent output object will get a value of missing in the observations added by the later output object. If a variable created by an output object has a different type than a variable with the same name created by an earlier output object, it will be added to the output data set using a new name formed by adding a numeric suffix.

Examples

Example 1: Creating a Combined Output Data Set

ODS features:
- ODS _ALL_ CLOSE statement
- ODS HTML statement:
  - BODY=
  - CONTENTS=
  - FRAME=
  - PAGE=
- ODS LISTING statement:
  - CLOSE
- ODS OUTPUT statement

Other SAS features:
- PROC FORMAT
- PROC PRINT
- PROC TABULATE
- KEEP= data set option

Program Description  This example routes two output objects that PROC TABULATE produces to both the OUTPUT destination and the HTML destination. The result is two output objects that are combined by the ODS OUTPUT statement to create an output data set formatted as HTML output by the ODS HTML statement.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program

Create the input data set. The data set ENERGY contains data on expenditures of energy for business and residential customers in individual states in the Northeast and West regions of the United States.

```sas
data energy;
  length State $2;
  input Region Division state $ Type Expenditures @@;
```
& datalines;
1 1 ME 1 708 1 1 ME 2 379 1 1 NH 1 597 1 1 NH 2 301
1 1 VT 1 353 1 1 VT 2 188 1 1 MA 1 3264 1 1 MA 2 2498
1 1 RI 1 531 1 1 RI 2 358 1 1 CT 1 2024 1 1 CT 2 1405
1 2 NY 1 8786 1 2 NY 2 7825 1 2 NJ 1 4115 1 2 NJ 2 3558
1 2 PA 1 6478 1 2 PA 2 3695 4 3 MT 1 322 4 3 MT 2 232
4 3 ID 1 392 4 3 ID 2 298 4 3 WY 1 194 4 3 WY 2 184
4 3 CO 1 1215 4 3 CO 2 1173 4 3 NM 1 545 4 3 NM 2 578
4 3 AZ 1 1694 4 3 AZ 2 1448 4 3 UT 1 621 4 3 UT 2 438
4 3 NV 1 493 4 3 NV 2 378 4 4 WA 1 1680 4 4 WA 2 1122
4 4 OR 1 1014 4 4 OR 2 756 4 4 CA 1 10643 4 4 CA 2 10114
4 4 AK 1 349 4 4 AK 2 329 4 4 HI 1 273 4 4 HI 2 298
;

Format the variables Region, Division, and Type.  PROC FORMAT creates formats for Region, Division, and Type.

proc format;
  value regfmt 1='Northeast' 2='South' 3='Midwest' 4='West';
  value divfmt 1='New England' 2='Middle Atlantic' 3='Mountain' 4='Pacific';
  value usetype 1='Residential Customers' 2='Business Customers';
run;

Do not produce listing output. The ODS LISTING statement closes the LISTING destination to conserve resources. (Otherwise, output would be written to the LISTING destination by default.)

ods listing close;

Create the SAS output data set. The ODS OUTPUT statement creates the SAS data set energyoutput from the output objects that PROC TABULATE produces. The name of each output object is Table. You can determine the name of the output objects by using the ODS TRACE ON statement. For information about the ODS TRACE statement, see “ODS TRACE Statement” on page 197.

Specify the variables that you want to be written to the output SAS data set. The KEEP= data set option limits the variables in the output data set energyoutput to Region, Division, Type, and Expenditures_sum. The variable name Expenditures_sum is generated by PROC TABULATE to indicate that the sum statistic was generated for the Expenditures variable.

ods output Table=energyoutput(keep=region division type expenditures_sum);

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC TABULATE is sent to the body file. FRAME=, CONTENTS=, and PAGE= create a frame that includes a table of contents and a table of pages that link to the contents of the body file. The body file also appears in the frame.

ods html body='your_body_file.html'
  frame='your_frame_file.html'
Create output data sets and an HTML report. This PROC TABULATE step creates two output objects named Table, one for each BY group, and adds them to the energyoutput data set. Because the HTML destination is open, ODS writes the output to the body file.

```
proc tabulate data=energy format=dollar12.;
   by region;
   class division type;
   var expenditures;
   table division,
            type*expenditures;
   format region regfmt. division divfmt. type usetype.;
   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

Close the current body file and open a new file. The ODS HTML BODY= statement closes the original body file and opens a new one. The contents, page, and frame files remain open. The contents and page files will contain links to both body files.

```
ods html body='your_body_file_2.html';
```

Print the combined data set. This PROC PRINT step prints the data set energyoutput that contains both BY groups. The output is added to the current body file, your_body_file_2.html.

```
proc print data=energyoutput noobs;
   title 'Combined Output Data Set';
run;
```

Close all of the open destinations. The ODS _ALL_ CLOSE statement closes all open ODS output destinations. To return ODS to its default setup, the ODS LISTING statement opens the LISTING destination.

```
ods _all_ close;
ods listing;
```
HTML Output

Display 5.7 Combined Data Set

The following HTML output shows the output DATA set that is created by the ODS OUTPUT statement.

![Combined Output Data Set](image)

Display 5.8 Output Objects Created by PROC TABULATE

The following output shows the two separate BY groups that are created by the TABULATE procedure.
Example 2: Using Different Procedures to Create a Data Set from Similar Output Objects

ODS features:

ODS HTML statement:
  BODY=
  CONTENTS=
  FRAME=

ODS OUTPUT statement
ODS SELECT statement

Other SAS features:

PROC GLM
PROC PRINT
PROC REG

Data set: IRON on page 191

Program Description

This example creates and prints a data set that is created from the parameter estimates that PROC REG and PROC GLM generate. These procedures are part of SAS/STAT software.

Note: This example uses file names that may not be valid in all operating environments. To successfully run the example in your operating environment, you may need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program

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

```sas
options nodate pageno=1 pagesize=60 linesize=72;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The FRAME= and CONTENTS= options create a frame that includes a table of contents that links to the contents of the body file. The body file also appears in the frame.

```sas
ods html body='parameter-estimates-body.htm'
   frame='parameter-estimates-frame.htm'
   contents='parameter-estimates-contents.htm';
```

Specify the output objects to be sent to all open ODS destinations. The ODS SELECT statement specifies that output objects named ParameterEstimates should be sent to all open ODS destinations that do not specifically exclude them. The LISTING destination is open by default, and its default list is SELECT ALL. The ODS HTML statement has opened the HTML destination, and its default list is also SELECT ALL. Thus any object that is named ParameterEstimates will go to both these destinations. The PERSIST option specifies that ParameterEstimates should remain in the overall selection list until the list is explicitly modified.

```sas
ods select ParameterEstimates(persist);
```
Create the IronParameterEstimates data set. The ODS OUTPUT statement opens the OUTPUT destination and creates the SAS data set IronParameterEstimates. By default, the list for the OUTPUT destination is EXCLUDE ALL. This ODS OUTPUT statement puts ParameterEstimates in the selection list for the destination. The PERSIST=PROC option specifies that ParameterEstimates should remain in the overall selection list until the procedure ends or the list is explicitly modified.

```plaintext
ods output ParameterEstimates=IronParameterEstimates (persist=proc);
```

Create the output objects. PROC REG and PROC GLM each produce an output object named ParameterEstimates. Because the data set definition persists when the procedure ends, ODS creates a output object from each one.

```plaintext
proc reg data=iron;
    model loss=fe;
    title 'Parameter Estimate from PROC REG';
    run;
quit;

proc glm data=iron;
    model loss=fe;
    title 'Parameter Estimate from PROC GLM';
    run;
quit;
```

Allow all open destinations to receive output objects. The ODS SELECT ALL statement sets the lists for all destinations to their defaults so that ODS sends all output objects to the HTML and LISTING destinations. (Without this statement, none of the output objects from the following PROC PRINT steps would be sent to the open destinations.)

```plaintext
ods select all;
```

Print the reports. The PROC PRINT steps print the data set that ODS created from PROC REG and PROC GLM. The output from these steps goes to both the HTML and the LISTING destinations. Links to the HTML output are added to the contents file.

```plaintext
proc print data=IronParameterEstimates noobs;
    title 'PROC PRINT Report of the Data set from PROC REG';
    run;
```

Close the OUTPUT and HTML destinations. The ODS _ALL_ CLOSE statement closes all open destinations except for the LISTING destination, which is open by default.

```plaintext
ods _all_ close;
```
**HTML Output**

**Display 5.9** HTML Output from the REG, GLM, and PRINT Procedures

The HTML output includes the parameter estimates from PROC REG, the parameter estimates from PROC GLM, and a report of the data set that ODS created from each set of parameter estimates.

The table of contents identifies output objects by their labels. The label for ParameterEstimates in PROC REG is Parameter Estimates. The corresponding label in PROC GLM is Solution. Notice how the column widths in the HTML output are automatically adjusted to fit the data. Compare this layout to the layout of the columns in the listing output.

**Listing Output**

**Output 5.3** Listing Output from the REG, GLM, and PRINT Procedures

<table>
<thead>
<tr>
<th>Parameter Estimate from PROC REG</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The REG Procedure</td>
<td></td>
</tr>
<tr>
<td>Model: MODELL1</td>
<td></td>
</tr>
<tr>
<td>Dependent Variable: Loss</td>
<td></td>
</tr>
<tr>
<td>Parameter Estimates</td>
<td></td>
</tr>
</tbody>
</table>

| Variable | DF  | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|----------|-----|--------------------|----------------|---------|---------|
| Intercept| 1   | 129.78660          | 1.40274        | 92.52   | <.0001  |
| Fe       | 1   | -24.01989          | 1.27977        | -18.77  | <.0001  |
Dictionary of ODS Language Statements  △  ODS OUTPUT Statement  145

Parameter Estimate from PROC GLM

| Parameter | Estimate  | Standard Error | t Value | Pr > |t| |
|-----------|-----------|----------------|--------|------|-------|
| Intercept | 129.7865993 | 1.40273671     | 92.52  | <.0001 |
| Fe        | -24.0198934  | 1.27976715     | -18.77 | <.0001 |

The GLM Procedure
Dependent Variable: Loss

PROC PRINT Report of the Data Set Created from PROC GLM and PROC REG

<table>
<thead>
<tr>
<th>Model</th>
<th>Dependent Variable</th>
<th>DF</th>
<th>Estimate</th>
<th>StdErr</th>
<th>tValue</th>
<th>Probt</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL1</td>
<td>Loss</td>
<td>1</td>
<td>129.78660</td>
<td>1.40274</td>
<td>92.52</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>MODEL1</td>
<td>Loss</td>
<td>1</td>
<td>-24.01989</td>
<td>1.27977</td>
<td>-18.77</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

Example 3: Creating a Data Set With and Without The MATCH_ALL Option

ODS features:
- ODS HTML statement:
  BODY=
- ODS LISTING
- ODS OUTPUT statement:
  MATCH_ALL
- ODS TRACE statement

Other SAS features:
- PROC PRINT
- PROC REG

Data set:
"Creating the Model Data Set" on page 621

Program Description  This example illustrates the differences in the data sets created by specifying the MATCH_ALL option and by not specifying the MATCH_ALL option. The first program creates a merged data set by specifying the MATCH_ALL option. The second program creates a merged data set without specifying the MATCH_ALL option. The data sets that are printed are parameter estimates that PROC REG generates. The PROC REG procedure is part of SAS/STAT software.

Note: This example uses file names that may not be valid in all operating environments. To successfully run the example in your operating environment, you may need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program 1

Do not create LISTING output. The ODS LISTING statement closes the LISTING destination to conserve resources. Otherwise, output would be written to the LISTING destination by default.

ods listing close;
Prepare a SAS data set to be created. The ODS OUTPUT statement opens the OUTPUT destination. By default, the list for the OUTPUT destination is EXCLUDE ALL. This ODS OUTPUT statement puts `SelectionSummary` in the selection list for the destination.

The MATCH_ALL option produces a SAS data set for each instance of `SelectionSummary`. The name of the first data set is `summary`, and the name of the second data set is `summary1`. ODS stores a list of these names in the macro variable `list`. This variable is used later in the example to combine the data sets.

```
ods output SelectionSummary(match_all=list) = summary;
```

Create the output objects and view a record of them in the log. PROC REG creates the output objects.

The ODS TRACE statement writes to the SAS log a record of each output object that is created. The ODS TRACE OFF statement represses the printing of the records.

```
ods trace on;
proc reg data=model;
  model r33=a b r4 r8 c d e r23 r24 r29/ selection=forward
      sle=.5 maxstep=3;
  model r33=a b r4 r8 c d e r23 r24 r29/ selection=backward
      sls=0.05 maxstep=3;
run;
ods trace off;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output.

```
ods html body='combined.html';
```

Print the reports. The PROC PRINT steps print the data sets that ODS created from PROC REG. The output from these steps is sent to both the HTML destination.

```
title2 'The First Data Set Has the VARENTERED Column';
proc print data=summary;
run;

title1;
title2 'The Second Data Set Has the VERREMOVED Column';
proc print data=summary1;
run;
```

Create a data set that contains all of the data sets. The DATA set `summarym` combines all the data sets that were created by the ODS OUTPUT statement. The macro variable `list` contains the list of data set names.

```
data summarym;
  set &list;
run;
```
Print the merged report and specify the title. The PROC PRINT step prints the merged data set created from the data step. The output from this step is sent to the HTML destination. The TITLE1 statement cancels the first title, and the TITLE2 statements specify a new title for the output.

```
title1;
title2 'The Merged Data Set Has Both Columns';
proc print data=summarym;
run;
```

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

```
ods html close;
```

**HTML Output**

Display 5.10 Three Data Sets Created When Using the MATCH_ALL option

**The First Data Set Created When Using the MATCH_ALL Option** This HTML output contains a printed report of the `summary` data set created by the ODS OUTPUT statement with the MATCH_ALL option specified. It has no `VERREMOVED` column.

**The Second Data Set Created When Using the MATCH_ALL Option** This HTML output contains a printed report of the `summary1` data set created by the ODS OUTPUT statement with the MATCH_ALL option specified. It has no `VARENTERED` column.

**The Merged Data Set Created When Using the MATCH_ALL Option** This HTML output contains a printed report of the `summarym` data set created by the ODS OUTPUT statement with the MATCH_ALL option specified. This is the data set created from `summary` and `summary1`. It contains both the `VARENTERED` and `VERREMOVED` columns.
Program 2

**Prepare a SAS data set to be created.** The ODS OUTPUT statement opens the OUTPUT destination and creates the SAS data set `Summary`. Because the MATCH_ALL option is not specified, ODS creates one data set that contains all instances of the output object `SelectionSummary`.

```
ods output SelectionSummary=summary;
  title1 'Without the MATCH_ALL Option, ODS Produces a Single Data Set With All Of the Columns';
```

**Create the output objects and view a record of them in the log.** PROC REG creates the output objects. The ODS TRACE statement writes to the SAS log a record of each output object that is created. The ODS TRACE OFF statement represses the printing of the records.

```
ods trace on;
  proc reg data=model;
    model r33=a b r4 r8 c d e r23 r24 r29/ selection=forward
        sle=.5 maxstep=3;
    model r33=a b r4 r8 c d e r23 r24 r29/ selection=backward
        sls=0.05 maxstep=3;
  run;
  ods trace off;
```

**Create HTML output.** The ODS HTML statement opens the HTML destination and creates HTML output.

```
ods html body='combined2.html';
```

**Print the combined data set.** The PROC PRINT step prints the merged data set created by ODS. The output from this step is sent to the HTML destination.

```
proc print data=summary;
  run;
```

**Close the HTML destination.** The ODS HTML CLOSE statement closes the HTML destination and all of the files that are associated with it.

```
ods html close;
```

**HTML Output**

**Display 5.11** Using the ODS OUTPUT Statement Without the MATCH_ALL Option to Combine Data Sets

This HTML output contains a printed report of the `summary` data set created by the ODS OUTPUT statement without the MATCH_ALL option specified. Note that to merge data sets, you do not have to specify the MATCH_ALL option.
**ODS PATH Statement**

Specifies locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them

**Valid:** anywhere  
**Category:** ODS: Output Control  
**Featured in:** Example 1 on page 281 and Example 2 on page 283  
**Tip:** This statement overrides the ODS PATH statement for the duration of a PROC TEMPLATE step.

**Syntax**

```
PATH <(APPEND) | (PREPEND) | (REMOVE) > location(s);
PATH path-argument;
```

**Required Arguments**

**location(s)**

specifies one or more locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them. ODS searches the locations in the order that they appear on the statement. It uses the first definition that it finds that has the appropriate access mode (read, write, or update) set.

Each location has the following form:

```
<libref>.item-store <(READ | UPDATE | WRITE)>
```

**<libref>.item-store**

identifies an item store to read from, to write to, or to update. If an item store does not already exist, then the ODS PATH statement will create it.

**<libref>.item-store <(READ | UPDATE | WRITE)***

specifies the access mode for the definition. The access mode is one of the following:

- **READ**
  
  provides read-only access.

- **WRITE**

  provides write access (always creating a new template store) as well as read access.

- **UPDATE**

  provides update access (creating a new template store only if the specified one does not exist) as well as read access.

**Default:** READ

**Default:**

- SASUSER.TEMPLAT (UPDATE)  
- SASHELP.TMPLMST (READ)

**Note:** SAS stores all the definitions that it provides in SASHELP.TMPLMST.
Interaction: You can use the PATH statement in a PROC TEMPLATE step to temporarily override the ODS PATH statement (see “PATH Statement” on page 276).

Tip: If you want to be able to ignore all definitions that you create, then keep them in their own item stores so that you can leave them out of the list of item stores that ODS searches.

**path-argument**

specifies the setting or displaying of the ODS path.

*path-argument* can be one of the following:

- **RESET**
  sets the ODS path to the default settings SASUSER.TEMPLAT (UPDATE) and SASHELP.TMPLMST (READ).

- **SHOW**
  displays the current ODS path.

- **VERIFY**
  sets the ODS path to include only templates supplied by SAS. VERIFY is the same as specifying ODS PATH SASHELP.TMPLMST (READ).

**Options**

(APPEND | PREPEND | REMOVE)

adds or removes one or more locations to a path.

- **APPEND**
  adds one or more locations to the end of a path. When you append a location to a path, all duplicate instances (same name and same permissions) of that item store are removed from the path. Only the last item store with the same name and permissions are kept.

- **PREPEND**
  adds one or more locations to the beginning of a path. When you prepend a location with update permissions to a path, all duplicate instances (same name and same permissions) of that item store are removed from the path. Only the first item store with the same name and permissions are kept.

- **REMOVE**
  removes one or more locations from a path.

Default: If you do not specify an APPEND, PREPEND, or REMOVE option, then the ODS PATH statement overwrites the complete path.

---

**ODS PCL Statement**

Opens, manages, or closes the PCL destination, which produces printable output for PCL (HP LaserJet) files

Valid: anywhere

Category: ODS: Third-Party Formatted
Interaction: By default, when you execute a procedure that uses the FORMCHAR system option (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+|---+=-\<>*";
```

---

**Syntax**

```
ODS PCL <(<ID=>identifier)> <action>;
ODS PCL <(<ID=>identifier)> <option(s)>;
```

---

**Without an Action or Options**

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

---

**Actions**

An action does one of the following:
- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

The following table lists the actions available for the ODS PCL statement. For complete descriptions of actions see “Actions” on page 160.

---

**Table 5.1 ODS PCL Action Summary Table**

<table>
<thead>
<tr>
<th>Task</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close the PCL destination and the file that is associated with it.</td>
<td>CLOSE</td>
</tr>
<tr>
<td>Select output objects for the PCL destination.</td>
<td>SELECT</td>
</tr>
<tr>
<td>Exclude output objects from the PCL destination.</td>
<td>EXCLUDE</td>
</tr>
<tr>
<td>Write to the SAS log the current selection or exclusion list for the PCL destination.</td>
<td>SHOW</td>
</tr>
</tbody>
</table>

---

**Options**

The following table lists the options that are available for the ODS PCL statement. For more detailed descriptions of these options, see “ODS PRINTER Statement” on page 159.
Table 5.2 ODS PCL Option Summary Table

<table>
<thead>
<tr>
<th>Task</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify whether or not background colors are printed in text.</td>
<td>BACKGROUND=</td>
</tr>
<tr>
<td>Apply a specified color scheme to your output.</td>
<td>COLOR=</td>
</tr>
<tr>
<td>Specify the number of columns to create on each page of output.</td>
<td>COLUMNS=</td>
</tr>
<tr>
<td>Specify the file to write to.</td>
<td>FILE=</td>
</tr>
<tr>
<td>Specify a scaling factor to apply to all the font sizes that do not have an explicit unit of measure.</td>
<td>FONTSCALE=</td>
</tr>
<tr>
<td>Open multiple instances of the same destination at the same time.</td>
<td>ID=</td>
</tr>
<tr>
<td>Specify that ODS use the printer drivers that SAS provides.</td>
<td>SAS</td>
</tr>
<tr>
<td>Control page breaks.</td>
<td>STARTPAGE=</td>
</tr>
<tr>
<td>Specify the style definition to use in writing the PCL output.</td>
<td>STYLE=</td>
</tr>
<tr>
<td>Insert text into your output.</td>
<td>TEXT=</td>
</tr>
<tr>
<td>For tables with multiple pages, ensure uniformity from page to page within a single table.</td>
<td>UNIFORM</td>
</tr>
</tbody>
</table>

Details

Opening and Closing the PCL Destination  You can modify an open PCL destination with many ODS PCL options. However, the FILE= and SAS options will automatically close the open destination that is referred to in the ODS PCL statement, and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

The ODS Printer Family of Statements  The ODS PCL statement is part of the ODS printer family of statements. Statements in the printer family open the PCL, PDF, PRINTER, or PS destination, producing output that is suitable for a high-resolution printer. The ODS PDF, ODS PRINTER, and ODS PS statements are also members of the ODS printer family of statements.

See Also

Statements:

“ODS PDF Statement” on page 153
“ODS PRINTER Statement” on page 159
“ODS PS Statement” on page 177
“The Third-Party Formatted Destinations” on page 27
“Commonly Used ODS Terminology” on page 21
**ODS PDF Statement**

Opens, manages, or closes the PDF destination, which produces PDF output, a form of output that is read by Adobe Acrobat Reader and other applications.

Valid: anywhere

Category: ODS: Third-Party Formatted

**CAUTION:**
The PDF driver that SAS uses does not recognize all Microsoft Windows fonts. You must enter any such fonts into the SAS registry in order for SAS to find them. For information about the SAS registry, see *SAS Language Reference: Concepts*.

---

**Syntax**

```
ODS PDF <(<ID=>identifier)> <action>;
ODS PDF <(<ID=>identifier)> <option(s)>;
```

---

**Without an Action or Options**

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

---

**Actions**

An *action* does one of the following:

- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

The following table lists the actions available for ODS PDF statement. For complete descriptions see “Actions” on page 160.

---

**Table 5.3 ODS PDF Action Summary Table**

<table>
<thead>
<tr>
<th>Task</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close the PDF destination and the file that is</td>
<td>CLOSE</td>
</tr>
<tr>
<td>associated with it.</td>
<td></td>
</tr>
<tr>
<td>Select output objects for the PDF destination.</td>
<td>SELECT</td>
</tr>
<tr>
<td>Exclude output objects from the PDF destination.</td>
<td>EXCLUDE</td>
</tr>
<tr>
<td>Write to the SAS log the current selection or</td>
<td>SHOW</td>
</tr>
<tr>
<td>exclusion list for the PDF destination.</td>
<td></td>
</tr>
</tbody>
</table>

---

**Options**

The following table lists the options that are available for the ODS PDF statement. For more detailed descriptions of these options, see “Options” on page 161.
### Table 5.4 ODS PDF Option Summary Table

<table>
<thead>
<tr>
<th>Task</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the root name for the anchor tag that identifies each output object in the current file.</td>
<td>ANCHOR=</td>
</tr>
<tr>
<td>Insert the text string that you specify as the author in the metadata of a file.</td>
<td>AUTHOR=</td>
</tr>
<tr>
<td>Specify whether or not background colors are printed in text.</td>
<td>BACKGROUND=</td>
</tr>
<tr>
<td>Specify a string to use as the first part of all references that ODS creates in the file.</td>
<td>BASE=</td>
</tr>
<tr>
<td>Specify whether or not to generate and display the list of bookmarks for a PDF file.</td>
<td>BOOKMARKLIST=</td>
</tr>
<tr>
<td>Control the generation of bookmarks in a PDF file.</td>
<td>BOOKMARKGEN=</td>
</tr>
<tr>
<td>Apply a specified color scheme to your output.</td>
<td>COLOR=</td>
</tr>
<tr>
<td>Specify the number of columns to create on each page of output.</td>
<td>COLUMNS=</td>
</tr>
<tr>
<td>Specify the file to write to.</td>
<td>FILE=</td>
</tr>
<tr>
<td>Specify a scaling factor to apply to all the font sizes that do not have an explicit unit of measure.</td>
<td>FONTSCALE=</td>
</tr>
<tr>
<td>Open multiple instances of the same destination at the same time.</td>
<td>ID=</td>
</tr>
<tr>
<td>Insert a string of keywords into the output file’s metadata.</td>
<td>KEYWORDS=</td>
</tr>
<tr>
<td>Control whether notes are added to a PDF file for items that are associated with the FLYOVER= style attribute.</td>
<td>PDFNOTE</td>
</tr>
<tr>
<td>Specify that ODS use the printer drivers that SAS provides.</td>
<td>SAS</td>
</tr>
<tr>
<td>Control page breaks.</td>
<td>STARTPAGE=</td>
</tr>
<tr>
<td>Specify the style definition to use in writing the PDF output.</td>
<td>STYLE=</td>
</tr>
<tr>
<td>Insert the text string that you specify as the subject in the metadata of a file.</td>
<td>SUBJECT=</td>
</tr>
<tr>
<td>Insert the text string that you specify as the title in the metadata of a file.</td>
<td>TITLE=</td>
</tr>
<tr>
<td>Insert text into your output.</td>
<td>TEXT=</td>
</tr>
<tr>
<td>For multi-page tables, provide uniformity from page to page within a single table.</td>
<td>UNIFORM</td>
</tr>
</tbody>
</table>

### Details

**The ODS Printer Family of Statements**

The ODS PDF statement is part of the ODS printer family of statements. Statements in the printer family open the PCL, PDF,
PRINTED, or PS destination, producing output that is suitable for a high-resolution printer. The ODS PCL, ODS PRINTER, and ODS PS statements are also members of the ODS printer family of statements.

**Opening and Closing the PDF Destination** You can modify an open PDF destination with many ODS PDF options. However, the FILE= and SAS options will automatically close the open destination that is referred to in the ODS PDF statement, and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

**Examples**

**Example 1: Opening Multiple Instances of the Same Destination at the Same Time**

ODS features:
- ODS PDF statement:
  - Options:
    - ID=
    - STYLE=
    - FILE=
- Other SAS features:
  - PROC FORMAT
  - PROC SORT
  - PROC REPORT
  - NOBYLINE|BYLINE system option
  - #BYVAL parameter in titles

This example opens multiple instances of the PDF destination to create PDF output. One instance uses the default style definition and the second instance uses the STYLE= option to specify the D3D style definition.

**Program**

Create the input data set. The data set `grain_production` contains data on the amounts of wheat, rice, and corn that five leading grain-producing nations produced during 1995 and 1996.

```sas
DATA grain_production;
  LENGTH Country $ 3 Type $ 5;
  INPUT Year Country $ Type $ Kilotons;
  DATALINES;
  1995 BRZ Wheat 1516
  1995 BRZ Rice 11236
  1995 BRZ Corn 36276
  1995 CHN Wheat 102207
  1995 CHN Rice 185226
  1995 CHN Corn 112331
  1995 IND Wheat 63007
  1995 IND Rice 122372
  1995 IND Corn 9800
  1995 INS Wheat .
  1995 INS Rice 49860
  1995 INS Corn 8223
  1995 USA Wheat 59494
```
1995 USA Rice 7888  
1995 USA Corn 187300  
1996 BRZ Wheat 3302  
1996 BRZ Rice 10035  
1996 BRZ Corn 31975  
1996 CHN Wheat 109000  
1996 CHN Rice 190100  
1996 CHN Corn 119350  
1996 IND Wheat 62620  
1996 IND Rice 120012  
1996 IND Corn 8660  
1996 INS Wheat 4  
1996 INS Rice 51165  
1996 INS Corn 8925  
1996 USA Wheat 62099  
1996 USA Rice 7771  
1996 USA Corn 236064  

Sort the data set. PROC SORT sorts the data first by values of Year, then by values of Country, and finally by values of Type.

proc sort data=grain_production;  
  by year country type;  
run;

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

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

Close the LISTING destination so that no listing output is produced. The LISTING destination is open by default. The ODS LISTING statement closes the LISTING destination to conserve resources. (If the destination were left open, then ODS would produce both Listing and PDF output.)

ods listing close;

Create two different PDF output files at the same time. The ODS PDF statement opens the PDF destination and creates PDF output.

The file grain-1.pdf is created by the first ODS PDF statement. Because no style definition is specified, the default style, styles.printer, is used.

The file grain-2.pdf is created by the second ODS PDF statement with the ID= option specified. The STYLE= option specifies that ODS use the style definition D3d. The ID= option gives this instance of the PDF destination the name d3dstyle. If you do not specify the ID= option, this ODS PDF statement will close the instance of the PDF destination that was opened by the previous ODS PDF statement and open a new instance of the PDF destination. The file grain-1.pdf will contain no output.

ods pdf file="grain-1.pdf";  
ods pdf (id=d3dstyle) style=D3d file="grain-2.pdf";
Suppress the default BY line, suppress the printing of the date, and use the BY value in a title. The NOBYLINE option suppresses the BY line. The #BYVAL specification inserts the current value of the BY variable year into the title.

```
options nobyline nodate;
title 'Leading Grain-Producing Countries';
title2 'for #byval(year)';
```

**Produce a report.** This PROC REPORT step produces a report on grain production. Each BY group produces a page of output.

```
proc report data=grain_production nowindows;
   by year;
   column country type kilotons;
   define country / group width=14 format=$cntry.;
   define type / group 'Type of Grain';
   define kilotons / format=comma12.;
   footnote 'Measurements are in metric tons.';
run;
```

Restore the BY line and clear the second title statement. The BYLINE option restores the BY line. The TITLE2 statement clears the second TITLE statement.

```
options byline;
title2;
```

**Produce a report that contains one table for each year.** The TABLE statement in this PROC TABULATE step has Year as the page dimension. Therefore, PROC TABULATE explicitly produces one table for 1995 and one for 1996.

```
proc tabulate data=grain_production format=comma12.;
   class year country type;
   var kilotons;
   table year,
      country*type, kilotons*sum='' / box=_page_ misstext='No data';
   format country $cntry.;
   footnote 'Measurements are in metric tons.';
run;
```

Close the open destinations so that you can view or print the output. The ODS PDF CLOSE statement closes the first instance of the PDF destination and all of the files that are associated with it. The ODS PDF (ID=d3dstyle) statement closes the second instance of the PDF destination and all of the files that are associated with it. You must close the destinations before you can view the output with a browser or before you can send the output to a physical printer.

```
ods pdf close;
ods pdf(id=d3dstyle) close;
```
PDF Output

Display 5.12  PDF Output Without Style

Display 5.13  PDF Output Using D3D Style

See Also

Statements:
“ODS PCL Statement” on page 150
“ODS PRINTER Statement” on page 159
“ODS PS Statement” on page 177
ODS PHTML Statement

Opens, manages, or closes the PHTML destination, which produces simple HTML output that uses twelve style elements and no class attributes

Valid: anywhere
Category: ODS: Third-Party Formatted

Syntax

ODS PHTML action;
ODS PHTML <file-specification(s)> <option(s)>;

Options

For a complete list of options, see the “ODS MARKUP Statement” on page 109.

Details

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

ODS PRINTER Statement

Opens, manages, or closes the PRINTER destination, which produces printable output

Valid: anywhere
Category: ODS: Third-Party Formatted

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option, (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly, because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```sas
OPTIONS FORMCHAR="|-----|+-----++|-----|--\<><";
```
**CAUTION:**
When you are producing PostScript output, verify that your online viewer or printer is set to use the same paper size as the value that is specified by the OPTIONS PAPERSIZE= statement. Otherwise, some parts of your output might appear to be missing.

---

**Syntax**

```odsprinter <(<ID=>identifier)> <action>;
odsprinter <(<ID=>identifier)> <option(s)>;
```

**Without an Action or Options**

If you use the ODS PRINTER statement in the UNIX, VMS, or OS/390 operating environments without an action or options, then it opens the PRINTER destination and creates PostScript output, unless otherwise configured by your system administrator.

If you use the ODS PRINTER statement in the Windows operating environment without an action or options, then it prints to the default Windows printer.

**Actions**

An *action* does one of the following:

- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

An *action* can be one of the following:

**CLOSE**

- closes the destination and the file that is associated with it. You cannot print the file until you close the destination.
  - **Tip:** When an ODS destination is closed, ODS does not send output to that destination. Closing an unneeded destination frees some system resources.

**EXCLUDE exclusion(s) | ALL | NONE**

- excludes output objects from the destination.
  - **Default:** NONE
  - **Restriction:** The destination must be open for this action to take effect.
  - **Main discussion:** “ODS EXCLUDE Statement” on page 90

**SELECT selection(s) | ALL | NONE**

- selects output objects for the destination.
  - **Default:** ALL
  - **Restriction:** The destination must be open for this action to take effect.
  - **Main discussion:** “ODS SELECT Statement” on page 188

**SHOW**

- writes the current selection or exclusion list for the destination to the SAS log.
  - **Restriction:** The destination must be open for this action to take effect.
  - **Tip:** If the selection or exclusion list is the default list (SELECT ALL), then SHOW also writes the entire selection or exclusion list.
See also: “ODS SHOW Statement” on page 197

Options

ANCHOR='anchor-name'
specifies the root name for the anchor tag that identifies each output object in the current file.

Each output object must have an anchor tag for the bookmarks to reference. The references, which are automatically created by ODS, point to the name of an anchor. Therefore, each anchor name in a file must be unique.

anchor-name
is the root name for the anchor tag that identifies each output object in the current file.

ODS creates unique anchor names by incrementing the name that you specify. For example, if you specify ANCHOR='tabulate', then ODS names the first anchor tabulate. The second anchor is named tabulate1; the third is named tabulate2, and so on.

Requirement: You must enclose anchor-name in quotation marks.

Alias: NAMED_DEST= | BOOKMARK=

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Tip: You can change anchor names as often as you want by submitting the ANCHOR= option in a valid statement anywhere in your program. Once you have specified an anchor name, it remains in effect until you specify a new one.

Tip: Specifying new anchor names at various points in your program is useful when you want to link to specific parts of your PRINTER output. Because you can control where the anchor name changes, you know in advance what the anchor name will be at those points.

AUTHOR= 'author-text'
inserts into the metadata of a file, the text string that you specify as the author.

author-text
is the text in the metadata of an open file that indicates the author.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Requirement: You must enclose author-text in quotation marks.

BACKGROUND=NO | YES
specifies whether or not background colors are printed in text.

NO
suppresses the printing of background colors in text.

Alias: NOBACKGROUND is an alias for BACKGROUND=NO.

YES
allows the printing of background colors in text.

Alias: BACKGROUND is an alias for BACKGROUND=YES

Default: YES


BASE='base-text'
specifies the text to use as the first part of all references that ODS creates in the output file.

base-text
is the text that ODS uses as the first part of all references that ODS creates in the file.

Consider this specification:

```
BASE='http://www.your-company.com/local-url/
```

In this case, ODS creates references that begin with the string http://
www.your-company.com/local-url/. The appropriate anchor-name completes the link.

**Restriction:** Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

**Requirement:** You must enclose base-text in quotation marks.

**BOOKMARKLIST=** **HIDE | NONE | SHOW**
specifies whether or not to generate and display the list of bookmarks for a PDF file.

**Note:** The generation of the bookmarks is not affected by the setting of this option. Bookmarks are generated by the BOOKMARKGEN= option. △

HIDE generates a list of bookmarks for your PDF file. The bookmarks are not automatically displayed when you open the PDF file.

NONE specifies not to generate a list of bookmarks for your PDF file.

---

**Alias:** NO | OFF

**Alias:** NOBOOKMARKLIST is an alias for BOOKMARKLIST=NONE | NO | OFF.

SHOW generates a list of bookmarks for your PDF file. The bookmarks are automatically displayed when you open the PDF file.

---

**Alias:** YES | ON

**Alias:** BOOKMARKLIST is an alias for BOOKMARKLIST=SHOW | YES | ON.

**Default:** SHOW

**Restriction:** This option can only be set when you first open the destination.

**Restriction:** This option only has an affect only when creating PDF or PDFMARK output.

**Interaction:** The NOTOC option specifies BOOKMARKLIST= OFF and CONTENTS= OFF.

**BOOKMARKGEN=** **NO | YES**
controls the generation of bookmarks in a PDF file.

NO does not generate bookmarks in the PDF file.

---

**Alias:** OFF

**Alias:** NOBOOKMARKGEN is an alias for BOOKMARKGEN=NO | OFF.

YES generates bookmarks in the PDF file.

---

**Alias:** ON

**Alias:** BOOKMARKGEN is an alias for BOOKMARKGEN=YES | ON.
**Default:** YES

**Restriction:** This option can only be set when you first open the destination.

**Interaction:** If you set BOOKMARKGEN=NO, then the BOOKMARKLIST option is set to NO also.

**COLOR=FULL | GRAY | MONO | NO | YES**

applies the specified color scheme to your output.

- **FULL**
  - creates full color output for both text and graphics.

- **GRAY**
  - creates grayscale output for both text and graphics.
    
    **Alias:** GREY

- **MONO**
  - creates monochromatic output for both text and graphics.
    
    **Alias:** BW

- **NO**
  - does not use all the color information that the style definition provides.

  **Interaction:** Specifying COLOR=NO is the same as specifying COLOR=GRAY and BACKGROUND=NO.

  **Tip:** If you specify COLOR=NO, then the destination does this:
  
  - generates black and white output
  - creates all text and rules in black
  - sets the SAS/GRAPH device to produce SAS/GRAPH output in grayscale
  - ignores specifications for a background color from the style definition except for the purposes of determining whether to print rules for the table

- **YES**
  - uses all the color information that a style definition provides, including background color.

  **Interaction:** Specifying COLOR=YES is the same as specifying COLOR=FULL and BACKGROUND=YES.

**Default:** YES

**Tip:** If you choose color output for a printer that does not support color, then your output might be difficult to read.

**Tip:** In order to actually print in color, you must also

- use a printer that is capable of printing in color
- use the COLORPRINTING SAS system option. For information about the COLORPRINTING system option, see *SAS Language Reference: Dictionary*.

**COLUMNS=n**
specifies the number of columns to create on each page of output.

- **n**
  - is the number columns per page.

**Default:** 1

**COMPRESS=n**
controls the compression of a PDF file. Compression reduces the size of the file.
n specifies the level of compression. The larger the number, the greater the compression. For example, \( n=0 \) is completely uncompressed, and \( n=9 \) is the maximum compression level.

**Default:** 6  
**Range:** 0–9  
**Restriction:** Use this option only with the ODS PDF statement and the ODS PRINTER statement with the PDF option specified.

**CONTENTS= NO | YES**

**CAUTION:**  
CONTENTS= is an experimental option. Do not use this option in production jobs.

controls the generation of a printable table of contents.

NO  
does not generate a printable table of contents.  
**Alias:** NOCONTENTS is an alias for CONTENTS=NO

YES  
generates a printable table of contents.  
**Alias:** CONTENTS is an alias for CONTENTS=YES

**FILE='external-file' | fileref**

specifies the file that contains the output.

*external-file*

is the name of an external file to write to.  
**Requirement:** You must enclose *external-file* in quotation marks.

*fileref*

is a fileref that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

**See:** For information about the FILENAME statement, see SAS Language Reference: Dictionary.

**Default:** If you do not specify a file to write to, then ODS writes to the file that is specified by one of two SAS system options:

**SYSPRINT=**  
if you are using the Windows operating environment and do not specify any of the following options: PCL, PDF, PDFMARK, PS, or SAS.

**PRINTERPATH=**  
in all other cases  
If the system option does not specify a file, then ODS writes to the default printer. For more information, see the PRINTER= option on page 166.

**Interaction:** In an ODS printer family statement that refers to an open ODS printer destination, the FILE= option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see “Opening and Closing the Printer Destination” on page 170.

**See:** For information about the FILENAME statement, see SAS Language Reference: Dictionary.

**FONTSCALE=percent**

specifies a scaling factor to apply to all the font sizes that do not have an explicit unit of measure.
percent
is the percent specified. Some SAS style definitions specify the font size as an
integer between 1 and 7. When ODS encounters such definitions, the PRINTER
destination arbitrarily selects a font size for each integer.

Default: 100

Restriction: FONTSCALE= has no effect unless it is used in combination with the
STYLE= option and a style definition that does not specify units of measure.

HOST
specifies that ODS use the printer drivers that the host system provides.

Interaction: In an ODS printer family statement that refers to an open ODS
printer destination, the HOST option forces ODS to close the destination and all
files that are associated with it, and to open a new instance of the destination. For
more information, see “Opening and Closing the Printer Destination” on page 170.

(<ID=> identifier)
enables you to open multiple instances of the same destination at the same time.
Each instance can have different options.

identifier
can be numeric or can be a series of characters that begin with a letter or an
underscore. Subsequent characters can include letters, underscores, and numerals.

Restriction: If identifier is numeric, it must be a positive integer.

Requirement: The ID= option must be specified immediately after the destination
name.

KEYWORDS='keywords-text'
inserts into the output file’s metadata, a string of keywords. The keywords enable a
document management system to do topic-based searches.

keywords-text
is the string of keywords.

Restriction: Use this option only with the ODS PDF statement, the ODS PS
statement with the PDFMARK option specified, and the ODS PRINTER statement
with the PDFMARK option specified.

Requirement: You must enclose keywords-text in quotation marks.

NOTOC
specifies that ODS omit the table of contents (Bookmark list) that is produced by
default when producing PDF or PDFMARK output.

Interaction: The NOTOC option specifies BOOKMARKLIST=OFF and
CONTENTS= OFF.

PCL
creates PCL output.

Restriction: Do not use this option in conjunction with the PDF or PS option.

Interaction: If you use the PCL option in an ODS PRINTER statement that refers
to an open ODS printer destination, the option will force ODS to close the
destination and all files that are associated with it, and to open a new instance of
the destination. For more information, see “Opening and Closing the Printer
Destination” on page 170.
PDF
creates PDF output.

**Restriction:** Do not use this option in conjunction with the PCL or PS options.

**Interaction:** If you use the PDF option in an ODS PRINTER statement that refers to an open ODS printer destination, the option will force ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see “Opening and Closing the Printer Destination” on page 170.

**PDFMARK**
enables ODS to insert special tags into a PostScript file. When you use software such as Adobe Acrobat (not Adobe Viewer), Acrobat Distiller interprets the tags to create a PDF file that contains the following items:

- bookmarks for each section of the output and for each table.
- references for items that are associated with the URL= style attribute.
- notes for items that are associated with the FLYOVER= style attribute. Notes are optional, and are based on the PDFNOTE option.
- author, keywords, subject, and title in the metadata of a file.

**Default:** Because using PDFMARK implies PostScript output, SAS automatically uses the PostScript driver that SAS supplies with this option.

**Restriction:** You cannot use the PRINTER= option with the PDFMARK option.

**Requirement:** To create a PDF file, you must use specialized software, such as Adobe Acrobat Distiller to convert the marked-up PostScript file into a PDF formatted file.

**Interaction:** In an ODS printer family statement that refers to an open ODS printer destination, the PDFMARK option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see “Opening and Closing the Printer Destination” on page 170.

**Tip:** Use this option only if you plan to distill the output. Otherwise, it uses excess resources and does not enhance the results.

**PDFNOTE | NOPDFNOTE**
controls whether notes are added to a PDF file for items that are associated with the FLYOVER= style attribute.

**PDFNOTE**
adds notes to a PDF file for items that are associated with the FLYOVER= style attribute.

**NOPDFNOTE**
modifies the behavior of PDFMARK so that notes are not added to the file for items that are associated with the FLYOVER= style attribute.

**Default:** PDFNOTE

**Restriction:** Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and ODS PRINTER statement with the PDFMARK option specified.

**PRINTER='printer-name'**
creates output that is formatted for the specified printer.

**printer-name**
is the name of the printer for which you want output formatted.
**Requirement:** You must enclose `printer-name` in quotation marks.

**Restriction:** `printer-name` must match a subkey in either the SAS registry or the Windows printer registry.

**Tip:** The description of the printer includes its destination and device type. If you are using the SAS printer drivers, then you can find a description of the printer in the `CORE PRINTING PRINTERS printer PRINTER SETUP OUTPUT` directory. If you are using the Windows operating environment and you do not specify the SAS option in the ODS PRINTER statement, then a description of the printer is located in the Windows registry.

**Note:** `printer-name` is not necessarily a physical printer. It is a description that tells SAS how to format the output, and where the output is located. For example, it could be a file on a disk.

**Alias:** PRT

**Default:** If you do not specify a printer, then ODS formats the printer output for the printer that is specified by one of two SAS system options:

- If you are using the Windows operating environment and do not specify any of the following options: PCL, PDFMARK, POSTSCRIPT, PS, or SAS.
- If the system option does not specify a printer, then ODS writes to the default printer driver as specified in the SAS registry or the Windows registry.

If the system option does not specify a printer, then ODS writes to the default printer driver as specified in the SAS registry, the default printer is specified in `CORE PRINTING Default Printer` directory.

**Restriction:** You cannot use the PRINTER= option with the PCL, PDF, PDFMARK, or PS options.

**Interaction:** In an ODS printer family statement that refers to an open ODS printer destination, the PRINTER= option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see “Opening and Closing the Printer Destination” on page 170.

**Tip:** To see a list of available printers for SAS printing, use the REGEDIT command. The printers are listed in the Registry Editor window under `CORE PRINTING PRINTERS`.

**PS**

creates PostScript output.

**Alias:** POSTSCRIPT

**Restriction:** Do not use this option in conjunction with the PDF or PCL options.

**Tip:** Specifying this option is equivalent to specifying both the SAS option and PRINTER= POSTSCRIPT.

**Interaction:** If you use the PS option in an ODS PRINTER statement that refers to an open ODS printer destination, the option will force ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see “Opening and Closing the Printer Destination” on page 170.
SAS specifies that ODS use the printer drivers that the SAS system provides.

**Interaction:** In an ODS printer family statement that refers to an open ODS printer destination, the SAS option forces ODS to close the destination and all files that are associated with it, and to open a new instance of the destination. For more information, see “Opening and Closing the Printer Destination” on page 170.

**STARTPAGE=NEVER | NO | NOW | YES** controls page breaks.

- **NEVER** specifies not to insert page breaks, even before graphics procedures.
  
  **CAUTION:** Each graph normally requires an entire page. The default behavior forces a new page after a graphics procedure, even if you use STARTPAGE=NO. STARTPAGE=NEVER turns off that behavior, so specifying STARTPAGE= NEVER might cause graphics to overprint.

- **NO** specifies that no new pages be inserted at the beginning of each procedure, or within certain procedures, even if new pages are requested by the procedure code. A new page will begin only when a page is filled or when you specify STARTPAGE=NOW.

  **CAUTION:** Each graph normally requires an entire page. The default behavior forces a new page after a graphics procedure, even if you use STARTPAGE=NO. STARTPAGE=NEVER turns off that behavior.

  **Alias:** OFF

- **NOW** forces the immediate insertion of a new page.

  **Tip:** This option is useful primarily when the current value of the STARTPAGE= option is NO. Otherwise, each new procedure forces a new page automatically.

- **YES** inserts a new page at the beginning of each procedure, and within certain procedures, as requested by the procedure code.

  **Alias:** ON

**Default:** YES

**STYLE=style-definition** specifies the style definition to use in writing the printer output.

**Default:** If you do not specify a style definition, then ODS uses the style definition that is specified in the SAS registry subkey:

```
ODS DESTINATIONS PRINTER Selected Style
```

By default, this value is `styles.printer`.

**Main discussion:** For a complete discussion of style definitions, see “The Default Style Definition for HTML and Markup Languages” on page 320.

**See also:** For instructions on making your own user-defined style definitions, see Chapter 10, “TEMPLATE Procedure: Creating Tabular Output,” on page 367.
SUBJECT='subject-text'
inserts into the metadata of a file, the text string that you specify as the subject.

subject-text
is the text in the metadata of a file that indicates the subject.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Requirement: You must enclose subject-text in quotation marks.

TEXT='text-string'
inserts a text string into your output.

text-string
is the text that you want to insert into your output.

Requirement: You must enclose text-string in quotation marks.

Tip: If you are submitting more than one procedure step and you do not specify the STARTPAGE=NO option, each procedure will force a new page before the output. Therefore, any text that you specify with TEXT= will be on the same page as the previous procedure.

TITLE='title-text'
inserts into the metadata of a file, the text string that you specify as the title.

title-text
is the text in the metadata of a file that indicates the title.

Restriction: Use this option only with the ODS PDF statement, the ODS PS statement with the PDFMARK option specified, and the ODS PRINTER statement with the PDFMARK option specified.

Requirement: You must enclose title-text in quotation marks.

UNIFORM
for multiple page tables, ensures uniformity from page to page within a single table. When the UNIFORM option is in effect, ODS reads the entire table before it starts to print it so that it can determine the column widths that are necessary to accommodate all the data. These column widths are applied to all pages of a multiple page table.

Note: With BY-group processing, SAS writes the results of each BY-group to a separate table, so the output might not be uniform across BY-groups.

Default: If you do not specify the UNIFORM option, then ODS prints a table one page at a time. This approach ensures that SAS does not run out of memory while processing very large tables. However, it can also mean that column widths vary from one page to the next.

Tip: The UNIFORM option can cause SAS to run out of memory if you are printing a very large table. If this happens, then you can explicitly set the width of each of the columns in the table, and then print the table one page at a time. To do so, you must edit the table definition that you use. For more information, see “What Can You Do with a Table Definition?” on page 368.
Details

Opening and Closing the Printer Destination  You can modify an open PRINTER destination with many ODS PRINTER options. However, any of the following options will automatically close the open destination that is referred to in the ODS PRINTER statement, and will also close any files that are associated with it, and then open a new instance of the destination: FILE=, HOST, PCL, PDF, PDFMARK, PRINTER=, PS, or SAS. If you use one of these options, it is best if you explicitly close the destination yourself.

For example, in the following ODS program, the second ODS PRINTER statement closes the PRINTER destination that is opened by the first ODS PRINTER statement. Therefore, the file `brickstyle.ps` will not contain output that is formatted with the `d3d` style. However, the second ODS PRINTER statement does not affect the PS destination that is opened by the ODS PS statement. The PS destination is still open and the file `nostyle.ps` could be modified.

The ODS PRINTER statement opens the PRINTER destination and creates PostScript output.

```ods printer ps style=brick file='brickstyle.ps';
proc print data=statepop;
run;
```

The ODS PS statement opens the PS destination and creates PostScript output.

```ods ps file='nostyle.ps';
proc print data=statepop;
run;
```

The ODS PRINTER statement closes the open PRINTER destination and the files that are associated with it. It then opens a new instance of the PRINTER destination and creates PostScript output.

```ods printer ps style=d3d file='d3dstyle.ps';
proc print data=statepop;
run;
ods printer ps close;
ods ps close;
```

Printing Output Directly to a Printer  Printing output directly to a printer using the ODS PRINTER statement depends on your host operating environment.

Note: To print directly to a printer in the OS/390, UNIX, or VMS operating environment, you can use the FILENAME statement. Specific information about your operating environment is required when using the FILENAME statement. See the SAS documentation for your operating environment before using this statement. Commands are also available in some operating environments that associate a fileref with a file and that break that association.
To send output to a printer when running SAS under this platform

<table>
<thead>
<tr>
<th>Platform</th>
<th>Instructions</th>
</tr>
</thead>
</table>
| OS/390   | Use the `FILENAME` statement with the `SYSOUT=` DATA set option specified. You can then print to the fileref.  
Syntax:  
```
FILENAME your-fileref sysout=a dest=printer-name;
ODS PRINTER file=your-fileref;
```
Example:  
```
FILENAME local sysout=a dest=chpljj21;
ODS PRINTER file=local;
``` |
| UNIX     | Use the `FILENAME` statement with the `PIPE` command to associate a fileref with your `lpr` print command.  
Syntax:  
```
FILENAME your-fileref pipe 'lpr -P printer-name';
ODS PRINTER file=your-fileref;
```
Example:  
```
FILENAME local pipe 'lpr -p chpljj21';
ODS PRINTER file=local;
``` |
| VMS      | Use the `FILENAME` statement with the `PRINTER` device type specified to create a printer fileref that you can print to.  
Syntax:  
```
FILENAME your-fileref printer passall=yes queue=printer-name;
ODS PRINTER file=your-fileref;
```
Example:  
```
FILENAME local printer passall=yes queue=chpljj21;
ODS PRINTER file=local;
``` |
| Windows  | If you want to print to your default printer use this code.  
Syntax:  
```
ODS PRINTER;
```  
If you want to print to a printer that is not the default, then use the `PRINTER=` option to specify the printer name.  
Syntax:  
```
ODS PRINTER printer=printer-name;
```
Example:  
```
ODS PRINTER printer=chpljj21;
``` |

**Note:** To find out what printers are available, select **Start -> Settings -> Printers** from the Taskbar. If a printer is listed there, then you can use it with the ODS PRINTER statement. If the printer name has spaces, then you must put the printer name in quotation marks.

---

**Using ODS PRINTER with Windows**  
When you use the ODS PRINTER statement in the Windows operating environment, ODS will produce output that is formatted for your default Windows printer unless you specify a different printer by using the `PRINTER=` option on page 166. You can also produce printable output files in PCL, PDF, or PostScript format by using the appropriate option.
Using ODS PRINTER with All Other Hosts  When you use the ODS PRINTER statement in any other operating environment, ODS uses the SAS drivers to produce output files in PCL, PDF, or PostScript formats. By default, the ODS PRINTER statement produces PostScript output files. You can also produce printable output files in PCL or PDF format by using the appropriate option or registry setting.

Example

Example 1: Selecting Output for the HTML and PRINTER Destinations

ODS features:

ODS _ALL_ CLOSE
ODS HTML statement:
   BODY=
ODS PRINTER statement:
   FILE=
PS
ODS LISTING statement:
   CLOSE
ODS SELECT statement:
   with label
   with name
   with path

Other SAS features:

PROC UNIVARIATE

Data set:

“Creating the Statepop Data Set” on page 620

This example selects three output objects from a UNIVARIATE procedure step to send to both the HTML destination and to the PRINTER destination.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program

Prevent listing output from being created. The ODS LISTING statement closes the LISTING destination in order to conserve resources.

&ods listing close;

Set the SAS system options. The OPTIONS statement controls several aspects of the PRINTER output. The NODATE system option specifies that SAS not print the date and the time. The NONUMBER system option specifies that SAS not print the page number on the first title line of each page of SAS output. These options do not affect the HTML output.

&options nodate nonumber;
Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. BODY= sends all output objects to the external file that you specify. Some browsers require an extension of HTM or HTML on the filename.

```
ods html body='your_file.html';
```

Create PostScript output. The ODS PRINTER statement opens the PRINTER destination and the PS option specifies PostScript output. FILE= sends all output objects to the external file that you specify.

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

Specify the output objects to send to the open destinations. The ODS SELECT statement specifies three output objects to send to all open destinations. The first output object is selected by its name, BasicMeasures. The second output object is selected by its label, Tests For Location. These two selection criteria select the output objects for the analysis of both variables. The third output object is selected by its full path, Univariate.CityPop_90.ExtremeObs. This selection criterion selects the output object for only one variable, CityPop_90.

```
ods select BasicMeasures 'Tests For Location'
                   Univariate.CityPop_90.ExtremeObs;
```

Compute descriptive statistics for two variables. PROC UNIVARIATE computes descriptive statistics for two variables, CityPop_80 and CityPop_90. ODS routes the selected output objects to the HTML and PRINTER destinations.

```
proc univariate data=statepop mu0=3.5;
  var citypop_90 citypop_80;
run;
```

Close the open destinations so that you can view or print the output. The ODS _ALL_ CLOSE statement closes all of the open destinations and all of the files that are associated with them. You must close the destinations before you can view the output with a browser, or before you can send the output to a physical printer.

```
ODS _all_ close;
```

Reset the default output type to LISTING. The ODS LISTING statement opens the LISTING destination to return ODS to its default setup.

```
ods listing;
```
HTML Output

Display 5.14  HTML Output for the Variables CityPop_90 and CityPop_80

The HTML output includes three output objects for the variable CityPop_90, and two output objects for the variable CityPop_80.
Printer Output

Display 5.15  Partial PostScript Output for the Variables CityPop_90 and CityPop_80

The Printer output includes three output objects for the variable CityPop_90, and two output objects for the variable CityPop_80.

The SAS System

The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

<table>
<thead>
<tr>
<th>Basic Statistical Measures</th>
<th>Location</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3.87020</td>
<td>5.15455</td>
</tr>
<tr>
<td>Median</td>
<td>2.42300</td>
<td>26.67064</td>
</tr>
<tr>
<td>Mode</td>
<td></td>
<td>28.65500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.80000</td>
</tr>
</tbody>
</table>

Tests for Location: Mu=3.5

<table>
<thead>
<tr>
<th>Test</th>
<th>Statistic</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student’s t</td>
<td>0.521924</td>
<td>0.5044</td>
</tr>
<tr>
<td>Sign</td>
<td>-9.5</td>
<td>0.0010</td>
</tr>
<tr>
<td>Signed Rank</td>
<td>-147</td>
<td>0.1798</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extreme Observations</th>
<th>Lowest</th>
<th>Highest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Value</td>
<td>Obs</td>
</tr>
<tr>
<td>0.184</td>
<td>41</td>
<td>9</td>
</tr>
<tr>
<td>0.132</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>0.191</td>
<td>39</td>
<td>26</td>
</tr>
<tr>
<td>0.221</td>
<td>36</td>
<td>7</td>
</tr>
<tr>
<td>0.228</td>
<td>50</td>
<td>49</td>
</tr>
</tbody>
</table>

ODS PROCLABEL Statement

Enables you to change a procedure label
Valid:  anywhere
Category:  ODS: Output Control
Interaction:  This statement applies to all open destinations, except for the output destination where a procedure label is not an option. However, this setting lasts for only one procedure step. You must issue an ODS PROCLABEL statement for each procedure step that you have.
Syntax

**ODS PROCTITLE**

```
ODS PROCTITLE | NOPROCTITLE;
```

**ODS PROCTITLE**

writes, in the output, the name of the procedure that produces the results.

*Note:* Not all procedures use a procedure title.

*Default:* ODS PROCTITLE is the default.

**ODS NOPROCTITLE**

suppresses the writing of the title of the procedure that produces the results.

Details

The following table lists the aliases for the ODS PROCTITLE statement:
ODS PS Statement

Opens, manages, or closes the PS destination, which Produces PostScript (PS) output

Valid: anywhere

Category: ODS: Third-Party Formatted

Interaction: By default, when you execute a procedure that uses the FORMCHAR system option, (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly, because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```sas
OPTIONS FORMCHAR="|----|+|---+=-|\<>*";
```

CAUTION:
When you are producing PostScript output, verify that your online viewer or printer is set to use the same paper size as the value that is specified by the OPTIONS PAPERSIZE= statement. Otherwise, some parts of your output might appear to be missing.

Syntax

```sas
ODS PS (<ID=>identifier> <action>);
ODS PS (<ID=>identifier> <option(s>));
```

Without an Action or Options

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

Actions

An action does one of the following:

- closes the destination
- excludes output objects
- selects output objects
The following table lists the actions that are available for the ODS PS statement. For complete descriptions of actions see “Actions” on page 160.

**Table 5.5 ODS PS Action Summary Table**

<table>
<thead>
<tr>
<th>Task</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close the PS destination and the file that is associated with it.</td>
<td>CLOSE</td>
</tr>
<tr>
<td>Select output objects for the PS destination.</td>
<td>SELECT</td>
</tr>
<tr>
<td>Exclude output objects from the PS destination.</td>
<td>EXCLUDE</td>
</tr>
<tr>
<td>Write to the SAS log the current selection or exclusion list for the PS destination.</td>
<td>SHOW</td>
</tr>
</tbody>
</table>

The following table lists the options available for the ODS PS statement. For more detailed descriptions of these options, see “Options” on page 161.

**Table 5.6 ODS PS Option Summary Table**

<table>
<thead>
<tr>
<th>Task</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the root name for the anchor tag that identifies each output object in the current file.</td>
<td>ANCHOR=</td>
</tr>
<tr>
<td>Insert the text string that you specify as the author in the metadata of a file.</td>
<td>AUTHOR=</td>
</tr>
<tr>
<td>Specify whether or not background colors are printed in text.</td>
<td>BACKGROUND=</td>
</tr>
<tr>
<td>Specify a string to use as the first part of all references that ODS creates in the file.</td>
<td>BASE=</td>
</tr>
<tr>
<td>Specify whether or not to generate and display the list of bookmarks for a PS file.</td>
<td>BOOKMARKLIST=</td>
</tr>
<tr>
<td>Control the generation of bookmarks in a PS file.</td>
<td>BOOKMARKGEN=</td>
</tr>
<tr>
<td>Apply a specified color scheme to your output.</td>
<td>COLOR=</td>
</tr>
<tr>
<td>Specify the number of columns to create on each page of output.</td>
<td>COLUMNS=</td>
</tr>
<tr>
<td>Specify the file to write to.</td>
<td>FILE=</td>
</tr>
<tr>
<td>Specify a scaling factor to apply to all the font sizes that do not have an explicit unit of measure.</td>
<td>FONTSCALE=</td>
</tr>
<tr>
<td>Open multiple instances of the same destination at the same time.</td>
<td>ID=</td>
</tr>
</tbody>
</table>
## Details

The ODS PS statement is part of the ODS printer family of statements. Statements in the printer family open the PCL, PDF, PRINTER, or PS destination, producing output that is suitable for a high-resolution printer. The ODS PCL, ODS PDF, and ODS PRINTER statements are also members of the ODS printer family of statements.

### Opening and Closing the PS Destination

You can modify an open PS destination with many ODS PS options. However, the FILE=, PDFMARK, and SAS options will automatically close the open destination that is referred to in the ODS PS statement and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

### See Also

- “ODS PCL Statement” on page 150
- “ODS PDF Statement” on page 153
- “ODS PRINTER Statement” on page 159
- “Commonly Used ODS Terminology” on page 21
- “The Third-Party Formatted Destinations” on page 27

## ODS RESULTS Statement

Tracks ODS output in the Results window

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

Syntax
ODS RESULTS ON | OFF;

Arguments
ON
Tracks output that is generated by ODS in the Results window.
OFF
Turns off the tracking of output that is generated by ODS 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 don’t want to track all of the output.

ODS RTF Statement
Opens, manages, or closes the RTF destination, which produces output written in Rich Text Format for use with Microsoft Word 2000
Valid: anywhere
Category: ODS: Third-Party Formatted
Interaction: By default, when you execute a procedure that uses the FORMCHAR system option, (for example, PROC PLOT or PROC CHART), ODS formats the output in SAS Monospace font. If you are creating output that will be viewed in an operating environment where SAS software is not installed, this output will not display correctly, because without SAS, the SAS Monospace font is not recognized. To make your document display correctly, include the following statement before your SAS program:

```
OPTIONS FORMCHAR="|----|+-----=|-/\*-";
```

Syntax
ODS RTF <(<ID=> identifier)> action;
ODS RTF <(<ID=> identifier)> <option(s)>;

Actions
An action does one of the following:
- closes the destination
- excludes output objects
- selects output objects
- writes the current exclusion list or selection list to the SAS log

An action can be one of the following:

**CLOSE**
closes the RTF 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 output objects from the RTF destination.
**Restriction:** The destination must be open for this action to take effect.
**Default:** NONE
**See also:** “ODS EXCLUDE Statement” on page 90

**SELECT** selection(s) | ALL | NONE
selects output objects for the RTF destination.
**Default:** ALL
**Restriction:** The destination must be open for this action to take effect.
**See also:** “ODS SELECT Statement” on page 188

**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.
**See also:** “ODS SHOW Statement” on page 197
**Tip:** If the selection or exclusion list is the default list (SELECT ALL), then SHOW
also writes the entire selection or exclusion list.

**Options**

**ANCHOR= ’anchor-name’**
specifies the base name for the RTF anchor tag that identifies each output object in
the current file.

Each output object must have an anchor tag for the contents, page, and frame files
to link to or to reference. The references, which are automatically created by ODS,
point to the name of an anchor. Therefore, each anchor name in a file must be unique.

anchor-name
is the base name for the RTF anchor tag that identifies each output object in the
current file.

ODS creates unique anchor names by incrementing the name that you specify.
For example, if you specify ANCHOR= ’tabulate’, then ODS names the first anchor
tabulate. The second anchor is named tabulate1; the third is named
tabulate2, and so on.
**Requirement:** You must enclose anchor-name in quotation marks.

**Alias:** NAMED_DEST= | BOOKMARK=

**Tip:** Specifying new anchor names at various points in your program is useful when
you want other RTF files to link to specific parts of your RTF output. Because you
can control where the anchor name changes, you know in advance what the anchor
name will be at those points.
Tip: You can change anchor names as often as you like by submitting the ANCHOR= option in an ODS RTF statement anywhere in your program. Once you have specified an anchor name, it remains in effect until you specify a new one.

AUTHOR= 'author-text'
inserts into the metadata of a file, the text string that you specify as the author.

author-text
is the text in the metadata of an open file that indicates the author.

Requirement: You must enclose author-text in quotation marks.

BASE= 'base-text'
specifies the text to use as the first part of references which ODS creates in the output file

base-text
is the text that ODS uses as the first part of all references that ODS creates in the file.

Consider this specification:

BASE='http://www.your-company.com/local-url'/

In this case, ODS creates links that begin with the string http://www.your-company.com/local-url/.

Requirement: You must enclose base-text in quotation marks.

COLUMNS= n
specifies the number of columns to create on each page of output.

n
is the number of page columns.

Default: 1

ENCODING= local-character-set-encoding
overrides the encoding for input or output processing (transcodes) of external files.


FILE= 'external-file' | 'fileref'
opens the RTF destination and specifies the RTF file or SAS catalog to write to. This file remains open until you do one of the following actions:

- close the RTF destination with ODS RTF CLOSE or ODS _ALL_ CLOSE
- specify another file to write to instead.

external-file
is the name of an external file to write to.

Requirement: You must enclose external-file in quotation marks.

fileref
is a fileref that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See also: For information about the FILENAME statement, see the section on statements in SAS Language Reference: Dictionary.

Requirement: You must enclose fileref in quotation marks.

Alias: BODY=

Interaction: In an ODS RTF statement that refers to an open RTF destination, the FILE= option forces ODS to close the destination and all files that are associated
with it, and to open a new instance of the destination. For more information, see “Opening and Closing the RTF Destination” on page 186.

See also: NEWFILE=

**FONTSCALE=** *percent*

specifies a scaling factor to apply to all the font sizes that do not have an explicit unit of measure.

*percent*

is the percent specified. Some SAS style definitions specify the font size as an integer between 1 and 7. When ODS encounters such definitions, the RTF destination arbitrarily selects a font size for each integer.

**Default:** 100

**Restriction:** FONTSIZE= has no effect unless it is used in combination with the STYLE= option.

**GFOOTNOTE | NOGFOOTNOTE**

controls the location of the footnotes that are defined by the graphics program that generates the RTF output.

**GFOOTNOTE**

includes all the currently defined footnotes within the graphics output.

**NOGFOOTNOTE**

suppresses all the currently defined footnotes from appearing in the graphics file. Instead, they become part of the RTF file.

**Default:** GFOOTNOTE

**Restriction:** This option applies only to SAS programs that produce one or more graph outputs.

**GTITLE | NOGTITLE**

controls the location of the titles that are defined by the graphics program that generates the RTF output.

**GTITLE**

includes all the currently defined titles within the graphics output that is called by the body file.

**NOGTITLE**

suppresses all the currently defined titles from appearing in the graphics output. Instead, they become part of the RTF file.

**Default:** GTITLE

**Restriction:** This option applies only to SAS programs that produce one or more graph files.

**(ID=** *identifier**)

*identifier*

can be a number, or a series of characters that begin with a letter or an underscore.

**Restriction:** If *identifier* is a number, it must be a positive.

**Requirement:** The ID= option must be specified immediately after the destination name.

**Tip:** You can omit the ID= option, and instead use a name or a number to identify the instance.

**Featured in:** Example 1 on page 155
KEEPN | NOKEEPN
controls how tables split at pages.

KEEPN
- ODS does not allow a table to split at a page break unless the entire table cannot fit on one page.

NOKEEPN
- ODS allows a table to split at a page break.

Tip: Although KEEPN minimizes page breaks in tables, it might use substantially more paper than NOKEEPN because it issues a page break before starting to print any table that does not fit on the remainder of the page.

NEWFILE= starting-point
creates a new file at the specified starting-point.
starting-point can be one of the following:

BYGROUP
- starts a new file for the results of each BY group.

NONE
- writes all output to the next file that is opened, and then stops incrementing.

OUTPUT
- starts a new file for the results of each BY group.

Alias: TABLE

PROC
- starts a new file each time that you start a new procedure.

Default: NONE
- ODS automatically names new files by incrementing the name of the body file. For example, if you specify FILE= 'REPORT.RTF', then ODS names the first file REPORT.RTF. Additional files are named REPORT1.RTF, REPORT2.RTF, and so on.
- If you end the file name with a number, then ODS begins incrementing with that number. For example, if you specify FILE= 'MAY5.RTF', then ODS names the first file MAY5.RTF. Additional files are named MAY6.RTF, MAY7.RTF, and so on.

Restriction: The NEWFILE= and TEXT= options cannot be used together in the same ODS RTF statement. You must use a separate ODS RTF statement for each of these options.

NOGFOOTNOTE
See the description of GFOOTNOTE | NOFOOTNOTE in this section.

NOGTITLE
See the description of GTITLE | NOGTITLE in this section.

OPERATOR= 'text-string'
inserts into the metadata of the RTF file, the text you specify.

- text-string
  - is the text in the metadata of a file that indicates the author.

Requirement: You must enclose text-string in quotation marks.

RECORD_SEPARATOR= 'alternative-separator' | NONE
specifies an alternative record separator, which is a character or string that separates lines in the output files.

Different operating environments use different separator characters. If you do not specify a record separator, then the RTF files are formatted for the environment in which you run the SAS job. However, if you are generating files in one operating
environment for viewing in another operating environment that uses a different separator character, then you can specify a record separator that is appropriate for the target environment.

**alternative-separator**

represents one or more characters, in hexadecimal or ASCII format. For example, the following option specifies a record separator of a carriage-return character and a linefeed character (on an ASCII file system):

```
RECORD_SEPARATOR= '0D0A'x
```

**Requirement:** You must enclose **alternative-separator** in quotation marks.

**NONE**

produces RTF output that is appropriate for the environment in which you run the SAS job.

**Operating Environment Information:** In many operating environments, using a value of **NONE** is the same as omitting the **RECORD_SEPARATOR** option. △

**Operating Environment Information:** In a mainframe environment, by default, ODS produces a binary file that contains embedded record-separator characters. While this approach means that the file is not restricted by the line-length restrictions on ASCII files, it also means that if you view the file in an editor, then the lines are concatenated.

If you want to format the RTF files so that you can read them with an editor, then use **RECORD_SEPARATOR= NONE**. In this case, ODS writes one line of RTF at a time to the file. When you use a value of **NONE**, the logical record length of the file that you are writing to must be at least as long as the longest line that ODS produces. Otherwise, RTF might wrap to another line at an inappropriate place. △

**Alias:**

```
RECSEP=
RS=
```

**SASDATE**

writes to the RTF file that the time and the date that you submitted your SAS program, instead of the time that the RTF file was opened.

**Restriction:** You can only specify **SASDATE** when a new file is opened. If you specify the option at any other time, a warning message is written to the SAS log.

**STARTPAGE= YES | NO | NOW**

controls page breaks.

**YES**

inserts a new page at the start of each procedure and within certain procedures, as requested by the procedure code.

**Alias:** ON

**NO**

specifies that no new pages be inserted explicitly at the start of each procedure or within certain procedures, even if new pages are requested by the procedure code. A new page will begin only when a page is filled or when you specify **STARTPAGE= NOW**.

**Alias:** NEVER

**Tip:** This option prints only the first set of titles and the first set of footnotes to the RTF file.
NOW forces the immediate insertion of a new page.

**Tip:** This option is useful primarily when the current value of the STARTPAGE= option is NO. Otherwise, each new procedure forces a new page automatically.

**Tip:** Specifying STARTPAGE= NO suppresses forced page breaks. You can turn on forced page breaking again by specifying STARTPAGE= YES. You can insert a page break by specifying STARTPAGE=NOW when you want a page break.

**Default:** YES

**STYLE= ’style-definition’**

specifies the style definition to use in writing the RTF files.

**style-definition**

describes how to display the presentation aspects (color, font face, font size, and so on) of your SAS output. A style definition determines the overall appearance of the documents that use it. Each style definition is composed of style elements.

**Main discussion:** For a complete discussion of style definitions, see “Overview: ODS Style Definitions” on page 285.

**See also:** For instructions on making your own user-defined style definitions, see Chapter 9, “TEMPLATE Procedure: Creating a Style Definition,” on page 285.

**Default:** If you do not specify a style definition, then ODS uses the file that is specified in the SAS registry subkey:

```
ODS DESTINATIONS RTF Selected Style
```

By default, this value specifies *styles.RTF*.

**TEXT= ’text-string’**

inserts text into your RTF output.

**text-string**

is the text that you want to insert into your RTF output. You can also use TEXT= to annotate other output.

**Restriction:** The NEWFILE= and TEXT= options cannot be used together in the same ODS RTF statement. You must use a separate ODS RTF statement for each of these options.

**Requirement:** You must enclose *text-string* in quotation marks.

**TITLE= ’title-text’**

inserts into the metadata of a file, the text string that you specify as the title.

**title-text**

is the text in the metadata of a file that indicates the title.

**Requirement:** You must enclose *title-text* in quotation marks.

**TRANTAB= translation-table**

specifies the translation table to use when transcoding a file for output.

**See:** For information about the TRANTAB= option see *SAS National Language Support (NLS): User’s Guide*.

**Details**

**Opening and Closing the RTF Destination**

You can modify an open RTF destination with many ODS RTF options. However, the FILE= option will automatically close the
open destination that is referred to in the ODS RTF statement and will also close any files associated with it, and then open a new instance of the destination. If you use one of these options, it is best if you explicitly close the destination yourself.

**How Does RTF Format Output?** RTF produces output for Microsoft Word 2000. Although there are other applications that can read RTF files, the RTF output might not work successfully with them.

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

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

In short, when producing RTF output for input to Microsoft Word, SAS determines the horizontal measurement and Microsoft Word controls the vertical measurement. Because Microsoft Word can determine how much room there is on the page, your tables will display consistently even after you modify your RTF file.

*Note:* The creation of complex tables that contain a large number of observations can reduce system efficiencies and slow down processing time.

**ODS RTF and Graphics** ODS RTF produces output in “rich text format”, that supports three formats for graphics that can be read by MS Word.

<table>
<thead>
<tr>
<th>Format for graphics</th>
<th>Corresponding SAS graphics driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>emfblips</td>
<td>SASEMF</td>
</tr>
<tr>
<td>pngblips</td>
<td>PNG</td>
</tr>
<tr>
<td>jpegblips</td>
<td>JPEG</td>
</tr>
</tbody>
</table>

When no target device is specified, the default target is SASEMF. The SASEMF graphics device is used as the default when you specify a driver other than SASEMF, PNG, or JPEG.

You can also use the ACTIVEX, ACTXIMG, JAVAIMG graphics drivers to generate graphics in your RTF documents. The ACTIVEX driver generates an ActiveX control. The ACTXIMG and JAVAIMG drivers generate PNG files. For more information about graphics devices, see *SAS/GRAPH Reference, Volumes 1 and 2.*
ODS SELECT Statement

Specifies output objects for ODS destinations

Valid: anywhere

Category: ODS: Output Control

See Also: “ODS EXCLUDE Statement” on page 90

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

Syntax

ODS <ODS-destination> SELECT selection(s) | ALL | NONE;

Arguments

selection(s)

specifies output objects to add to a selection list. ODS sends the items in the selection list to all active ODS destinations. By default, ODS automatically modifies selection lists when a DATA step that uses ODS or a procedure step ends. For information about modifying these lists, see “Selection and Exclusion Lists” on page 34. For information about ending DATA and procedure steps, see the section on DATA Step Processing in SAS Language Reference: Concepts.

Each selection has the following form:

output-object <(PERSIST)>

output-object

specifies the output object to select.

To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that is produced. You can specify an output object as

- a full path. For example,

```plaintext
Univariate.City_Pop_90.TestsForLocation
```

is the full path of the output object.

- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is

```plaintext
Univariate.City_Pop_90.TestsForLocation
```

then the partial paths are:

```plaintext
City_Pop_90.TestsForLocation
TestsForLocation
```

- a label that is enclosed by quotation marks.

For example,
"The UNIVARIATE Procedure"

- a label path. For example, the label path for the output object is
  
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  
  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement. △

- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is
  
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  
  then the partial label paths are:
  
  "CityPop_90"."Tests For Location"
  "Tests For Location"

- a mixture of labels and paths.

- any of the partial path specifications, followed by a pound sign (#) and a number. For example, TestsForLocation#3 refers to the third output object that is named TestsForLocation.

See also: “ODS TRACE Statement” on page 197

(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

- any ODS EXCLUDE statement
- ODS SELECT NONE
- ODS SELECT ALL
- an ODS SELECT statement that applies to the same output object but does not specify PERSIST.

Requirement: You must enclose PERSIST in parentheses.

ALL

specifies that ODS send all of the output objects to the open destination.

Alias: ODS SELECT DEFAULT

Interaction: If you specify ALL without specifying a destination, ODS sets the overall list to SELECT ALL and sets all other lists to their defaults.

NONE

specifies that ODS does not send any output objects to the open destination.

Interaction: If you specify NONE without specifying a destination, ODS sets the overall list to SELECT NONE and sets all other lists to their defaults.

Tip: Using the NONE action is different from closing a destination. The output destination is still open, but ODS is restricting the output that it sends to the destination.

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

**ODS-destination**
specifies which ODS destination’s selection list to write to, where *ODS-destination* can be any valid ODS destination. For a discussion of ODS destinations, see “What Are the ODS Destinations?” on page 25.

**Default:** If you omit *ODS-destination*, ODS writes to the overall selection list.

**Tip:** To set the selection list for the Output destination to something other than the default, see the “ODS OUTPUT Statement” on page 135.

Example

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

ODS features:

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

  **ODS SHOW statement**

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

Other SAS features:

- PROC GLM
- PROC PRINT
- PROC PLOT

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, so the contents file that is generated by the ODS HTML statement shows which output objects are routed to both the HTML and the LISTING destinations.

**Note:** This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

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

Create the input data set. The data set IRON contains data from Draper and Smith (p. 98).* Thirteen specimens of 90/10 copper-nickel alloys were tested in a corrosion-wheel setup in order to examine corrosion. Each specimen had a specified iron content. The wheel was rotated in salt sea water at 30 ft/sec for 60 days. Weight loss was used to quantify the corrosion. FE represents the iron content, and LOSS denotes the weight loss in milligrams/square decimeter/day.

```
data iron;
  input Fe Loss @@;
datalines;
0.01 127.6 0.48 124.0 0.71 110.8 0.95 103.9
1.19 101.5 0.01 130.1 0.48 122.0 1.44 92.3
0.71 113.1 1.96 83.7 0.01 128.0 1.44 91.4
1.96 86.2
;
```

Create HTML output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from the procedures is sent to the file `odspersist-body.htm`. The FRAME=, CONTENTS=, and PAGE= options create the files `odspersist-frame.htm`, `odspersist-contents.htm`, and `odspersist-page.htm`, respectively. These files, together with the file `odspersist-body.htm`, create a frame that includes a table of contents and a table of pages that link to the contents of the body file.

```
ods html body='odspersist-body.htm'
  frame='odspersist-frame.htm'
  contents='odspersist-contents.htm'
  page='odspersist-page.htm'
;
```

Write the overall selection list to the SAS log. The ODS SHOW statement writes to the SAS log the overall list, which is set to SELECT ALL by default. See [1] in “SAS Log” on page 194.

```
ods show;
```

Specify the output objects that will be sent to the open destinations. The ODS SELECT statement determines which output objects ODS sends to the LISTING and HTML destinations. In this case, ODS sends all output objects that are named ParameterEstimates and all output objects that are labeled Type III Model ANOVA to the two destinations.

```
ods select ParameterEstimates
  "Type III Model ANOVA";
```

Write the modified overall selection list to the SAS log. The ODS SHOW statement writes to the SAS log the overall selection list, which now contains the two items that were specified in the ODS SELECT statement. See [2] in the “SAS Log” on page 194.

```
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 196. 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”.

```sas
proc glm data=iron;
   model loss=fe;
   title 'Parameter Estimates and Type III Model ANOVA';
run;
```

Write the overall selection list to the SAS log. PROC GLM supports run-group processing. Therefore, the RUN statement does not end the procedure, and ODS does not automatically modify the selection list. See [3] in the “SAS Log” on page 194.

```sas
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.

```sas
quit;
```

Write the current selection list to the SAS log. The ODS SHOW statement writes the current selection list to the SAS log. See [4] in the “SAS Log” on page 194.

```sas
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 196. The QUIT statement ends the procedure. Because the list uses the argument ALL, ODS does not automatically modify it when the PROC step ends.

```sas
proc glm data=iron;
   model loss=fe;
   title 'All Output Objects Selected';
run;
quit;
```

Modify the overall selection lists. This ODS SELECT statement modifies the overall selection list so that it sends all output objects that are named OverallANOVA, and all output objects that are labeled Fit Statistics, to both the HTML and LISTING destinations. The PERSIST option specifies that OverallANOVA should remain in the selection list when ODS automatically modifies it.

```sas
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 196.

```sas
proc glm data=iron;
   model loss=fe;
```
title 'OverallANOVA and Fitness Statistics';
run;

End the GLM procedure and automatically modify the selection list. When the QUIT statement ends the procedure, ODS automatically modifies the selection list. Because OverallANOVA was specified with the PERSIST option, it remains in the selection list. Because Fitness Statistics was not specified with the PERSIST option, ODS removes it from the selection list.

quit;

Write the current selection list to the SAS log. The ODS SHOW statement writes the current selection list to the SAS log. See [5] in the “SAS Log” on page 194.

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 196.

proc glm data=iron;
  model loss=fe;
  title 'OverallANOVA';
  title2 'Part of the Selection List Persists';
run;

End the GLM procedure and automatically modify the selection list. When the QUIT statement ends the procedure, ODS automatically modifies the selection list. Because OverallANOVA was specified with the PERSIST option, it remains in the selection list.

quit;

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

proc print data=iron;
  title 'The IRON Data Set';
run;

Reset all selection lists. This ODS SELECT statement resets all selection lists to their defaults.

ods select all;

Create the plots. As PROC PLOT creates and sends each output object to the Output Delivery System, ODS sends each one to the HTML and LISTING destinations because their lists and the overall list is set to SELECT ALL (the default).

proc plot data=iron;
  plot fe*loss='*' / vpos=25 ;
  label fe='Iron Content'
    loss='Weight Loss';
  title 'Plot of Iron Versus Loss';
run;
**End the PLOT procedure.** The QUIT statement ends the PLOT procedure. Because the list uses the argument ALL, ODS does not automatically modify the list when the PROC step ends.

```
quit;
```

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

```ods html close;
```

**SAS Log**

**Output 5.4** The ODS SHOW Statement Writes the Current Selection List to the SAS Log.

```
10 ods html body='odspersist-body.htm'
11 contents='odspersist-contents.htm'
12 frame='odspersist-frame.htm'
13 page='odspersist-page.htm';
NOTE: Writing HTML Body file: odspersist-body.htm
NOTE: Writing HTML Contents file: odspersist-contents.htm
NOTE: Writing HTML Pages file: odspersist-page.htm
NOTE: Writing HTML Frames file: odspersist-frame.htm
14 ods show;
Current OVERALL select list is: ALL [1]
15 ods select ParameterEstimates
16 "Type III Model ANOVA";
17 ods show;
Current OVERALL select list is: [2]
1. ParameterEstimates
2. "Type III Model ANOVA"
18 proc glm data=iron;
19 model loss=fe;
20 title 'Parameter Estimates and Type III Model ANOVA';
21 run;
22 ods show;
Current OVERALL select list is: [3]
1. ParameterEstimates
2. "Type III Model ANOVA"
23 quit;
NOTE: PROCEDURE GLM used:
   real time       x.xx seconds
   cpu time        x.xx seconds
24 ods show;
Current OVERALL select list is: ALL [4]
25 proc glm data=iron;
26 model loss=fe;
27 title 'All Output Objects Selected';
28 run;
29 quit;
NOTE: PROCEDURE GLM used:
   real time       x.xx seconds
   cpu time        x.xx seconds
```
ods select OverallANOVA(persist) "Fit Statistics";
proc glm data=iron;
    model loss=fe;
    title 'OverallANOVA and Fitness Statistics';
run;
quit;
NOTE: PROCEDURE GLM used:
   real time x.xx seconds
   cpu time x.xx seconds
ods show;
Current OVERALL select list is: [5]
1. OverallANOVA(PERSIST)
proc glm data=iron;
    model loss=fe;
    title 'OverallANOVA';
    title2 'Part of the Selection List Persists';
run;
quit;
NOTE: PROCEDURE GLM used:
   real time x.xx seconds
   cpu time x.xx seconds
proc print data=iron;
    title 'The IRON Data Set';
run;
NOTE: PROCEDURE PRINT used:
   real time x.xx seconds
   cpu time x.xx seconds
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;
HTML Output

Display 5.16 Contents File Produced by the ODS HTML Statement

The contents file shows which output objects from each procedure were sent to the open ODS destinations. You can see that no output was written to the HTML destination for PROC PRINT (because it 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.

See Also

Statements:
“ODS EXCLUDE Statement” on page 90
“ODS SHOW Statement” on page 197
“ODS TRACE Statement” on page 197

ODS SHOW Statement

Writes the specified selection or exclusion list to the SAS log

Valid: anywhere
Category: ODS: Output Control

Syntax

ODS <ODS-destination> SHOW;

Options

ODS-destination specifies which ODS destination's selection or exclusion list to write to the SAS log, where ODS-destination can be any valid ODS destination. For information about ODS destinations, see “What Are the ODS Destinations?” on page 25. For information on selection and exclusion lists, see “Selection and Exclusion Lists” on page 34.

Default: If you omit ODS-destination, ODS SHOW writes the overall selection or exclusion list.

See Also

Statements:

“ODS EXCLUDE Statement” on page 90
“ODS SELECT Statement” on page 188
“ODS TRACE Statement” on page 197

ODS TRACE Statement

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

Valid: anywhere
Category: ODS: Output Control
Default: OFF
Featured in: Example 3 on page 145
Syntax

ODS TRACE ON<option(s)>;
ODS TRACE OFF;

Arguments

OFF
  turns off the writing of the trace record.

ON
  turns on the writing of the trace record.
  Alias: OUTPUT

Options

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 most useful for users who are running a localized version of SAS because the labels are translated from English to the local language. The names and paths of output objects are not translated because they are part of the syntax of the Output Delivery System.

LISTING
  writes the trace record to the Listing destination, so that each part of the trace record immediately precedes the output object that it describes.

Details

Contents of the Trace Record  ODS produces an output object by combining data from the data component with a table definition. The trace record provides information about the data component, the table definition, and the output object. By default, the record that the ODS TRACE statement produces contains these items:

  Name
  is the name of the output object. You can use the name to reference this output object and others with the same name. For details on how to reference an output object, see “How Does ODS Determine the Destinations for an Output Object?” on page 35. For example, you could use this name in an ODS OUTPUT statement to make a data set from the output object, or you could use it in an ODS SELECT or an ODS EXCLUDE statement.
  Tip:  The name is the rightmost part of the path that appears in the trace record.

  Label
  briefly describes the contents of the output object. This label also identifies the output object in the Results window.
Data name
is the name of the data component that was used to create this output object. The
data name appears only if it differs from the name of the output object.

Data label
describes the contents of the data.

Template
is the name of the table definition that ODS used to format the output object. You
can modify this definition with PROC TEMPLATE. See the EDIT statement “EDIT
Statement” on page 373 for more information.

Path
is the path of the output object. You can use the path to reference this output
object. For example, you could use the path in the ODS OUTPUT statement to
make a data set from the output, or you could use it in an ODS SELECT or an
ODS EXCLUDE statement.

The LABEL option modifies the trace record by including the label path for the object
in the record. See the discussion of the LABEL option.

### Specifying an Output Object
Once you have determined which output objects your SAS program produces, you can specify the output objects in statements such as ODS EXCLUDE, ODS SELECT, and so on. You can specify an output object by using one of the following:

- a full path. For example,
  ```
  Univariate.City_Pop_90.TestsForLocation
  ```
  is the full path of the output object.

- a partial path. A partial path consists of any part of the full path that begins
  immediately after a period (.) and continues to the end of the full path. For
  example, if the full path is
  ```
  Univariate.City_Pop_90.TestsForLocation
  ```
  then the partial paths are:
  ```
  City_Pop_90.TestsForLocation
  TestsForLocation
  ```

- a label that is enclosed by quotation marks.
  For example,
  ```
  "The UNIVARIATE Procedure"
  ```

- a label path. For example, the label path for the output object is
  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```
  **Note:** The trace record shows the label path only if you specify the LABEL
  option in the ODS TRACE statement.

- a partial label path. A partial label path consists of any part of the label that
  begins immediately after a period (.) and continues to the end of the label. For
  example, if the label path is
  ```
  "The UNIVARIATE Procedure"."CityPop_90"."Tests For Location"
  ```
then the partial label paths are:

"CityPop_90"."Tests For Location"
"Tests For Location"

- a mixture of labels and paths.
- any of the partial path specifications, followed by a pound sign (#) and a number. For example, TestsForLocation#3 refers to the third output object that is named TestsForLocation.

**Example**

**Example 1: Determining Which Output Objects a Procedure Creates**

ODS features:

- ODS TRACE statement:
  - LABEL
  - OFF
  - ON
- Other SAS features:
  - PROC UNIVARIATE

Data set:

- STATEPOP ✩ Creating the Statepop Data Set ✩ on page 620

This example shows how to determine the names and labels of the output objects that a procedure creates. You can use this information to select and exclude output objects.

**Note:** This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

**Program**

- Specify that SAS write the trace record to the SAS log and include label paths. This ODS TRACE statement writes the trace record to the SAS log. The LABEL option includes label paths in the trace record.

```sas
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.

```sas
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.

```sas
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. On the other hand, 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_80.Moments
Label Path: "The Univariate Procedure"."CityPop_80"."Moments"
------------

Output Added:
------------
Name: BasicMeasures
Label: Basic Measures of Location and Variability
Template: base.univariate.Measures
Path: Univariate.CityPop_80.BasicMeasures
Label Path: "The Univariate Procedure"."CityPop_80"."Basic Measures of Location and Variability"
------------

Output Added:
------------
Name: TestsForLocation
Label: Tests For Location
Template: base.univariate.Location
Path: Univariate.CityPop_80.TestsForLocation
Label Path: "The Univariate Procedure"."CityPop_80"."Tests For Location"
------------

Output Added:
------------
Name: Quantiles
Label: Quantiles
Template: base.univariate.Quantiles
Path: Univariate.CityPop_80.Quantiles
Label Path: "The Univariate Procedure"."CityPop_80"."Quantiles"
------------

Output Added:
------------
Name: ExtremeObs
Label: Extreme Observations
Template: base.univariate.ExtObs
Path: Univariate.CityPop_80.ExtremeObs
Label Path: "The Univariate Procedure"."CityPop_80"."Extreme Observations"
------------

See Also

Statements:
"ODS EXCLUDE Statement" on page 90
"ODS SELECT Statement" on page 188

ODS USEGOPT Statement

Determines whether or not ODS uses graphics option settings

Valid: anywhere

Category: ODS: Output Control

See also: SAS/GRAPH Reference, Volumes 1 and 2
**Syntax**

`ODS USEGOPT | NOUSEGOPT;`

**Details**

**Enabling Graphics Options** While ODS USEGOPT is in effect, the settings for the following graphics options will affect all of your ODS output, including tables:

- `CTEXT=`
- `CTITLE=`
- `FTITLE=`
- `FTEXT=`
- `HTEXT=`
- `HTITLE=`

If ODS NOUSEGOPT is in effect, the settings for these graphics options will not override the value in the style definition for titles and footnotes in your ODS output.

**Examples**

**Example 1: Enabling and Disabling Graphics Options**

ODS features:

- ODS HTML statement:
  - `FILE=`
- ODS LISTING statement:
  - `CLOSE`
- ODS NOUSEGOPT statement
- ODS USEGOPT statement

Other SAS features:

- `GOPTIONS` statement:
  - `FCTEXT=`
  - `FTITLE=`
  - `HTEXT=`

PROC PRINT
- `TITLE` statement

Data set:
- `sashelp.class`

**Program Description** This example creates two HTML reports, one with the `GOPTIONS` enabled by using the ODS USEPGOT statement, and one with `GOPTIONS` disabled by using the ODS NOUSEGOPT statement.
**Program**

**Specify the GOPTIONS.** The RESET=ALL option sets all graphics options to their default values and cancels all global statements. The HTEXT= option specifies that the text height for titles and footnotes be two units. The FTITLE= option specifies the font for titles and footnotes. The FTEXT option specifies the font for the text.

```plaintext
goptions reset=all htext=2 ftitle=script ftext=script;
```

**Do not produce listing output.** The ODS LISTING statement closes the LISTING destination to conserve resources. Otherwise, output would be written to the LISTING destination by default.

```plaintext
ods listing close;
```

**Enable the graphics options.** While ODS USEGOPT is in effect, the settings for HTEXT= and CTEXT= graphics option will override values that are specified for titles and footnotes in the style definition.

```plaintext
ods usegopt;
```

**Create HTML output, specify titles, and print the data set.** The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file specified by the FILE= option. The TITLE statements specify the titles for your output. The PRINT procedure prints the SAS data set from the SASHELP library. The OBS= option specifies two observations to be printed.

```plaintext
ods html file="opts.html";
title "This Title Was Created With the USEGOPT Option Specified" ;
title2 "The Graphics Option Settings are Turned On";
proc print data=sashelp.class (obs=2);
run;
```

**Disable the graphics options.** The NOUSEGOPT statement suppresses the use of the HTEXT= and CTEXT= graphics option settings for your output.

```plaintext
ods nousegopt;
```

**Create HTML output, specify titles, and print the data set.** The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file specified by the FILE= option. The TITLE statements specify the titles for your output. The PRINT procedure prints the SAS data set from the SASHELP library. The OBS= option specifies two observations to be printed.

```plaintext
title "This Title Was Created With the NOUSEGOPT Option Specified" ;
title2 "The Graphics Option Settings are Turned Off";
```
proc print data=sashelp.class (obs=2) ;
run;

Close the HTML destination and open the LISTING destination. The ODS HTML CLOSE statement closes the HTML destination. To return ODS to its default setup, the ODS LISTING statement opens the LISTING destination.

ods html close;
ods listing;

Display 5.17  HTML Output

In the following example, the heights and fonts for the titles of the first table are specified by the FTITLE, FTEXT, and HTEXT options in the GOPTIONS statement. The heights and fonts for the titles of the second table are specified by the default style definition.

ODS VERIFY Statement

Prints or suppresses a message indicating that a style definition or a table definition being used is not supplied by SAS

Valid: anywhere
Category: ODS: Output Control

Syntax

ODS VERIFY <ON | OFF | ERROR | WARN>;
Options

ON | OFF | ERROR | WARN

ON
prints the warning and sends output objects to open destinations.
Alias: ODS VERIFY

OFF
suppresses the warning.
Alias: ODS NOVERIFY

ERROR
prints an error message instead of a warning message and does not send output objects to open destinations.

WARN
prints a warning message and does not send output objects to open destinations.

Default: If you do not specify the ODS VERIFY statement, then ODS runs with the verification process turned off. If you specify the ODS VERIFY statement but do not specify an argument, then ODS runs with verification turned on.

Tip: For information about how to ignore user-created definitions, see “ODS PATH Statement” on page 149.

Details

Why Use the ODS VERIFY Statement? PROC TEMPLATE can modify the values in an output object. None of the definitions that SAS provides modifies any values. If you receive a warning from the ODS VERIFY statement, then look at the source code to verify that the values have not been modified.

ODS WML Statement

Opens, manages, or closes the WML destination, which uses the Wireless Application Protocol (WAP) to produce a Wireless Markup Language (WML) DTD with a simple list for a table of contents

Valid: anywhere

Category: ODS: Third-Party Formatted

Syntax

ODS WML < (<ID=>identifier)> action;

ODS WML < (<ID=>identifier)> <file-specification(s)> <option(s)> <option(s)>;

Options

For a complete list of options, see the “ODS MARKUP Statement” on page 109.
Details

The ODS WML statement is part of the ODS markup family of statements. ODS statements in the markup family produce output that is formatted using one of many different markup languages, such as HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. You can specify a markup language that SAS supplies, or create one of your own and store it as a user-defined markup language.
PART 4

The DOCUMENT Procedure

Chapter 6. The DOCUMENT Procedure 211
CHAPTER 6

The DOCUMENT Procedure

Overview: DOCUMENT Procedure 212
Why Use the DOCUMENT Procedure? 212
DOCUMENT Procedure Terminology 212

Syntax: DOCUMENT Procedure 213
PROC DOCUMENT Statement 215
COPY Statement 216
DELETE Statement 217
DIR Statement 217
DOC Statement 218
DOC CLOSE Statement 219
HIDE Statement 219
IMPORT Statement 220
LINK Statement 221
LIST Statement 222
MAKE Statement 223
MOVE Statement 224
NOTE Statement 225
OBANOTE Statement 226
OBBNOTE Statement 227
OBFOOTN Statement 228
OBPAGE Statement 228
OBSTITLE Statement 229
OBTITLE Statement 230
RENAME Statement 231
REPLAY Statement 231
SETLABEL Statement 232
UNHIDE Statement 233

The DOCUMENT PROCEDURE and BY-Groups 233

Concepts: DOCUMENT Procedure 235
What Is an ODS Document? 235
Definition 235
What Does an ODS Document Include? 235
What Is Not Included in an ODS Document? 235
ODS Document Persistence 235
What Is an ODS Document Path? 236
Definition of ODS Document Path 236
Entry Names 236
Understanding Sequence Numbers 236
ODS Documents and Base SAS Procedures 236
Getting Familiar with Output Objects 237
How Do ODS Documents Interact across Operating Environments? 237
Overview: DOCUMENT Procedure

Why Use the DOCUMENT Procedure?

When your output is in an ODS document, the DOCUMENT procedure enables you to rearrange, duplicate, or remove output from the results of a procedure or a database query. Also, you can generate output for one or more ODS destinations using the newly transformed output hierarchy file. Thus, the DOCUMENT procedure enables you to

- transform your report without having to rerun your analysis or repeat your database query.
- have more control over the structure of your output.
- display your output to any ODS output format without executing your SAS programs again.
- navigate the current file location and list entries.
- open and list ODS documents.
- manage output.
- store the ODS output objects in raw form. The output is kept in the original internal representation as a data component plus a table definition.

The DOCUMENT procedure is an interactive procedure that enables you to use ODS and global statements within the PROC DOCUMENT step.

Unlike other ODS destinations, the DOCUMENT destination has a graphical user interface (GUI) for performing tasks. However, you can also do the same tasks with batch statement syntax using the DOCUMENT procedure.

DOCUMENT Procedure Terminology

current document is the open document.
current path is the location in the open document where you currently reside.
entry is one or more links, output objects, files, or partitioned data sets.
graph segment is a file type or output object that contains a graph. Graphs are created in some SAS procedures, including those in SAS/GRAPH. The graph output object is referenced as a GRSEG.

See: For more information about GRSEG and SAS/GRAPH procedures, see SAS/GRAPH Reference, Volumes 1 and 2.

path is the route through a hierarchal file system, leading to a particular file or file location of an entry. path refers to the physical location of an entry.

replay occurs when you generate your output again, in the same or different format, without rerunning your analyses or data queries.

root file location is the top level of a file location in an ODS document. A root file location is not contained within another file location and does not have a name assigned. A root file location is similar to the root directory of a Windows operating environment.

Syntax: DOCUMENT Procedure

PROC DOCUMENT <options>;
   COPY path <, path-2, ...path-n> TO path <option(s)>;
   DELETE path <, path-2, ...path-n>;
   DIR <path>;
   DOC <options>;
   DOC CLOSE;
   HIDE path <, path-2, ...path-n>;
   IMPORT DATA= data-set-name | GRSEG= grseg TO path <options>;
   LINK path TO path <options>;
   LIST <path-1, path-2, ...path-n> <option(s)>;
   MAKE path <, path-2, ...path-n> <options>;
   MOVE path <, path-2, ...path-n> TO path <option(s)>;
   NOTE path '<text'> <option(s)>;
   OBANOTE<n> output-object '<text'> <option>;
   OBBNOTE<n> output-object '<text'> <option>;
   OBFOOTN<n> output-object '<text'>;
   OBPAGE output-object <option(s)>;
   OBSTITLE<n> output-object '<text'> <option(s)>;
   OBTITLE<n> output-object '<text'>;
   RENAME path-1 TO path-2;
   REPLAY <path <, path-2, ...path-n>> <options>;
   SETLABEL path 'label';
   UNHIDE path <, path-2, ...path-n>;
   QUIT;
<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert a copy of an entry into a specified path</td>
<td>COPY</td>
</tr>
<tr>
<td>Delete entries from a specified path or paths</td>
<td>DELETE</td>
</tr>
<tr>
<td>Set or display the current directory</td>
<td>DIR</td>
</tr>
<tr>
<td>Open a document and its contents to browse or edit</td>
<td>DOC</td>
</tr>
<tr>
<td>Close the current document</td>
<td>DOC CLOSE</td>
</tr>
<tr>
<td>Prevent output from being displayed when the document is replayed</td>
<td>HIDE</td>
</tr>
<tr>
<td>Import a data set or graph segment into the current directory</td>
<td>IMPORT</td>
</tr>
<tr>
<td>Create a symbolic link from one output object to another output object</td>
<td>LINK</td>
</tr>
<tr>
<td>List the content of one or more entries</td>
<td>LIST</td>
</tr>
<tr>
<td>Create one or more new directories</td>
<td>MAKE</td>
</tr>
<tr>
<td>Move entries from one directory to another directory</td>
<td>MOVE</td>
</tr>
<tr>
<td>Create text strings in the current directory</td>
<td>NOTE</td>
</tr>
<tr>
<td>Create or modify lines of text after the specified output object</td>
<td>OBANOTE</td>
</tr>
<tr>
<td>Create or modify lines of text before the specified output object</td>
<td>OBBNOTE</td>
</tr>
<tr>
<td>Create or modify lines of text at the bottom of the page in which the output object is displayed</td>
<td>OBFOOTN</td>
</tr>
<tr>
<td>Create or delete a page break for an output object</td>
<td>OBPAGE</td>
</tr>
<tr>
<td>Create or modify subtitles</td>
<td>OBSTITLE</td>
</tr>
<tr>
<td>Create or modify lines of text at the top of the page where the output object is displayed</td>
<td>OBTITLE</td>
</tr>
<tr>
<td>Assign a different name to a directory or output object</td>
<td>RENAME</td>
</tr>
<tr>
<td>Replay one or more entries to the specified open ODS destination(s)</td>
<td>REPLAY</td>
</tr>
<tr>
<td>Assign a label to the current entry</td>
<td>SETLABEL</td>
</tr>
<tr>
<td>Enable the output of a hidden entry to be displayed when it is replayed</td>
<td>UNHIDE</td>
</tr>
</tbody>
</table>
PROC DOCUMENT Statement

Creates or opens a document to modify

Default: Documents are opened in the UPDATE access mode.

Caution: If you do not explicitly close the DOCUMENT destination with an ODS DOCUMENT CLOSE statement, then ODS continues to append files to your document.

PROC DOCUMENT <options <access-options>>;

Without Options

If no options are specified, then the PROC DOCUMENT statement opens the last document that was created in the current SAS session.

Options

NAME= <libref:memname <access-options>

specifies the name that you assign to a new or existing document and its access mode.

<libref:memname
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 valid SAS library member.

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

Default: UPDATE

READ
opens a document and provides read-only access.

Requirement: To open a document in the READ access mode, the document must already exist.

Interaction: If a label has been specified with the LABEL= option, then the label is ignored.

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

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

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

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

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

Caution: If the document already exists, then its contents will not be changed.
Interaction: If a label has been specified with the LABEL= option, then it will be assigned to the document.

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

LABEL='label'
assigns a label to your document.

Restriction: You can assign a label to your document only if you have write access permissions.

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

COPY Statement

Copies an entry into the specified path

Default: If you do not specify a location where to insert an entry into a path, then the entry is inserted at the end of the path.

COPY path <, path-2, ...path-n> TO path <\LEVELS= value | ALL >FIRST | LAST | BEFORE= path | AFTER= path>>;

Required Arguments

path
is the location where a link, output object, or file is copied.

Requirement: When you specify more than one path, you must separate the paths with a comma.

Options

AFTER= path
inserts a copy of an entry after the specified path.

BEFORE= path
inserts a copy of an entry before the specified path.

FIRST
inserts a copy of an entry at the beginning of the specified path.

LAST
inserts a copy of an entry at the end of the specified path.

LEVELS= ALL | value
specifies the depth of the file location.

Restriction: The LEVELS= option is a valid option only when you specify a file location.
ALL
  specifies all levels of the file location.

value
  specifies the numeric value of the file location level.

DELETE Statement

Deletes entries from the current file location

Restriction: You cannot delete or move the root file location.

Caution: The DELETE statement affects all levels of a file location below the specified path.

DELETE path <, path-2, ...path-n>;

Required Arguments

path
  specifies the location of one or more links, output objects, or file locations.
  Requirement: When you specify more than one path, you must separate the paths with a comma.

DIR Statement

Sets or displays the current file location

Featured In: Example 1 on page 244, Example 2 on page 248, and Example 3 on page 253

DIR <path>;

Without Options

If no options are specified, then the DIR statement displays the current path.

Options

path
  sets the current file location.
DOC Statement

Opens a document and its contents to browse or edit

Default: Documents are opened in the UPDATE access mode.

Featured In: Example 1 on page 244 and Example 2 on page 248

DOC <options <access-options>>;

Without Options

If no options are specified, then the DOC statement lists the ODS documents that exist in all SAS libraries.

Options

LABEL= 'label'
assigns a label to your document.

Restriction: You can assign a label to your document only if you have write access permission.

Requirement: To use the LABEL= option, you must specify the NAME= option on the DOC statement.

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

LIBRARY=library-name
specifies that only the documents in the specified library-name are listed.

Alias: LIB=

Interaction: You cannot specify the LIBRARY= option with the NAME= or LABEL= options.

NAME= libref.memname <access-options>
specifies the name that you assign to a document and its access mode.

<libref>memname
identifies a document.

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

Restriction: The document must be a valid 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 will be overwritten. If the document does not exist, then it will be created.

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

**UPDATE**
opens a document and provides update access, but only if you have update permission.

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

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

---

**DOC CLOSE Statement**

Closes the current document

```plaintext
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.

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

**Required Arguments**

*path*

specifies the location of the file or files that you want to hide.

**Requirement:** When you specify more than one path, separate the paths with a comma.
IMPORT Statement

Imports the specified SAS data set or graph segment to the current file location

**IMPORT DATA=** `data-set-name<data-set-options>` **|** **GRSEG=** `grseg` **TO** `path <l<br>
<FIRST | LAST | BEFORE= path | AFTER= path>>;`

**Required Arguments**

**DATA=** `data-set-name`

specifies an existing SAS data set that you want to import.

**GRSEG=** `grseg`

stores a reference to a graph segment.

`grseg`

specifies the 3-level catalog path name. For example,

GRSEG=SASUSER.grseg.mygraph.

See: GRSEG= option in the *SAS/GRAPH Reference, Volumes 1 and 2.*

`path`

specifies the location where you want to import the data set or graph segment.

**Options**

**AFTER=** `path`

imports the data set or graph segment into the file location after the specified path.

**BEFORE=** `path`

imports the data set or graph segment into the file location before the specified path.

**data-set-options**

specify actions that apply only to the SAS data set.

See also: For information about SAS data sets and their options, see *SAS Language Reference: Dictionary*

**FIRST**

imports the data set or graph segment at the beginning of the file location.

**LAST**

imports the data set or graph segment at the end the file location.
LINK Statement

Creates a symbolic link from one specified output object to another in the current file location

```
LINK path TO path / <HARD> <LABEL> <FIRST | LAST | BEFORE= path | AFTER= path>
```

Required Arguments

`path`

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

Options

```
AFTER= path
```

links to the entry that is after the specified path in the current file location.

```
BEFORE= path
```

links to the entry that is before the specified path in the current file location.

```
FIRST
```

links to the first entry in the current file location.

```
HARD
```

specifies a type of link that refers to a copy of an output object within the ODS document. All data is shared between the link and the target, except names and labels.

**Restriction:** A hard link can only reference 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. If you delete a hard link, you do not effect the target. Similarly, if you delete a target, you do not affect the link.

```
LABEL
```

copies the source label to the link.

**Default:** The source label is not copied unless you use the LABEL option.

```
LAST
```

links to the last entry in the current file location.
LIST Statement

Lists the contents of one or more entries

Default: If you omit the DETAILS option, then only summary information is displayed.

Default: If you omit the ORDER= option, then the contents of the specified entries are listed in INSERT order (the order in which you arranged the entries.)

Tip: To see any entries that might be hidden in the current file location, use the LIST statement.

Featured In: Example 1 on page 244, Example 2 on page 248, and Example 3 on page 253

```plaintext
LIST <path-1, path-2, ...path-n> </DETAILS><FOLLOW><LEVELS= value | ALL><ORDER= ALPHA | DATE | INSERT>>;
```

Required Arguments

`path`

specifies the location of an entry. An entry can be one or more file locations, links, or output objects.

Requirement: When you specify more than one path, separate the paths with a comma.

Options

DETAILS

specifies the properties of the entries.

FOLLOW

resolves all links and lists the contents of the entries.

LEVELS= value | ALL

specifies the depth of the file locations that you want to list.

Default: If you omit the LEVELS= option, then the default value of the level is 1.

Restriction: The LEVELS= option is a valid option only when you specify a file location.

`value`

specifies the numeric value of the file location level.

ALL

specifies all levels of the file location.

ORDER= ALPHA | DATE | INSERT

specifies the order in which the entries are listed.

ALPHA

lists the entries in alphabetical order.
DATE
lists the file locations in order of ascending date/time stamp when the entries were created.

INSERT
lists the file locations in the order in which you arranged the entries.

**MAKE Statement**

**Creates one or more new file locations**

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

```
MAKE path <, path-2, ...path-n> < FIRST | LAST | BEFORE=path | AFTER=path >>;
```

**Required Arguments**

`path`
specifies the newly created file location.

**Requirement:** When you specify more than one path, separate the paths with a comma.

**Options**

*AFTER= path*
adds the newly created file location after the specified path in the current file location.

*BEFORE= path*
adds the newly created file location before the specified path in the current file location.

*FIRST*
adds the newly created file location to the beginning of the current file location.

*LAST*
adds the newly created file location to the end of the current file location.
**MOVE Statement**

Moves entries from the specified location to another location

**Restriction:** You can not move or delete the root file location.

**Requirement:** When you specify more than one path, separate the paths with a comma.

**Caution:** The MOVE statement effects all levels of a file location below the specified starting level.

\[
\text{MOVE } \textit{path} <, \textit{path} \cdot 2, \ldots \textit{path} \cdot n> \text{ TO } \textit{path} \cdot /<\text{LEVELS}= \textit{value} | \text{ALL} ><\text{FIRST} | \text{LAST} | \text{BEFORE}= \textit{path} | \text{AFTER}= \textit{path}>>;
\]

**Required Arguments**

\textit{path}

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

**Options**

\texttt{AFTER}= \textit{path}

moves the entry after the specified entry in the path.

\texttt{BEFORE}= \textit{path}

moves the entry before the specified entry in the path.

\texttt{FIRST}

moves the entry to the beginning of the specified file location.

\texttt{LAST}

moves the entry to the end of the specified file location.

\texttt{LEVELS}= \textit{value} | \texttt{ALL}

specifies the level in the file hierarchy.

\textit{value}

specifies the numeric value of the file location level. For example, “3” indicates the third level in the hierarchy.

\texttt{ALL}

specifies all levels of the file location.
NOTE Statement

Creates text strings in the current file location

Default: If you omit the JUST= option, then the note is centered between the left and right margins.
Default: If no location is specified, then the note is added to the end of the current location.
Featured In: Example 3 on page 253

NOTE path <'text'> <JUST= LEFT | CENTER | RIGHT> <FIRST | LAST | BEFORE= path | AFTER= path>>;

Without Options

If no text is specified, then the NOTE statement creates a blank note.

Required Arguments

path
specifies the location where the note is stored.

Options

AFTER= path
inserts the text string after the specified path.
BEFORE= path
inserts the text string before the specified path.
FIRST
inserts the text string at the beginning of the path.
JUST= LEFT | CENTER | RIGHT
specifies the alignment of the text string.
LEFT
aligns the text string with the left margin.
CENTER
aligns the text string in the center between the left and right margins.
RIGHT
aligns the text string with the right margin.
LAST
inserts the text string at the end of the path.

Requirement: All text strings must be enclosed in quotation marks.
OBANOTE Statement

Creates or modifies an object footer (lines of text) after the specified output object.

Featured In: Example 3 on page 253

```
OBANOTE<n> output-object <'object-footer-text'> </JUST= LEFT | CENTER | RIGHT>;
```

**Required Arguments**

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

**Options**

`JUST= LEFT | CENTER | RIGHT`

specifies the alignment of the object-footer.

- LEFT
  - aligns the object-footer-text with the left margin.

- CENTER
  - aligns the object-footer-text in the center between the left and right margins.

- RIGHT
  - aligns the object-footer-text with the right margin.

`n`

specifies the relative line that contains the note.

**Default:** If you omit `n`, SAS assumes a value of 1. Therefore, you can specify `OBANOTE` or `OBANOTE1` for the first text line.

**Range:** 1–10

**Tip:** The OBANOTE line with the highest number appears on the bottom line.

**Tip:** You can create notes that contain blank lines between them. For example, if you specify text with an OBANOTE1 statement that is followed by an OBANOTE3 statement, then a blank line separates the two lines of text.

`object-footer-text`

specifies the text string.

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 text, you can embed the items at the position where you want them to appear. For more information, see “The DOCUMENT PROCEDURE and BY-Groups” on page 233.

**Requirement:** All object-footer-text must be enclosed in quotation marks.

**Caution:** If no object-footer-text is specified, then the OBANOTE statement deletes all existing footer notes for the specified output object only.
OBBO NOTE Statement

Creates or modifies an object header (lines of text) before the output object

Featured In: Example 3 on page 253

```plaintext
OBBO NOTE<n> output-object <object-header-text> <JUST= LEFT | CENTER | RIGHT>
```

Required Arguments

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

Options

`JUST= LEFT | CENTER | RIGHT`
specifies the alignment of the object-header-text.

- LEFT
  - aligns the object-header-text with the left margin.
- CENTER
  - aligns the object-header-text in the center between the left and right margins.
- RIGHT
  - aligns the object-header-text with the right margin.

`n`
specifies the relative line that contains the note.

Default: If you omit `n`, SAS assumes a value of 1. Therefore, you can specify OBBO NOTE or OBBO NOTE1 for the first text line.

Range: 1–10

Tip: The OBBO NOTE line with the highest number appears on the bottom line.

Tip: You can create notes that contain blank lines between them. For example, if you specify text with an OBBO NOTE statement that is followed by an OBBO NOTE3 statement, then a blank line separates the two lines of text.

`object-header-text`
specifies the text string.

You can customize object headers by inserting BY variable values (#BYVALn), BY variable names (#BYVARn), or BY lines (#BYLINE) into object headers that are specified in PROC DOCUMENT steps. After you specify the object header text, you can embed the items at the position where you want them to appear. For more information, see “The DOCUMENT PROCEDURE and BY-Groups” on page 233.

Requirement: All object-header-text must be enclosed in quotation marks.

Caution: If no object-header-text is specified, then the OBBO NOTE statement deletes all existing header notes for the specified output object only.
OBFOOTN Statement

Creates or modifies lines of text at the bottom of the page on which the output object is displayed

Restriction:  You can print up to ten lines of text.
Tip:  The OBFOOTN statement is similar to the global FOOTNOTE statement.
Featured In:  Example 3 on page 253

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

Required Arguments

`output-object`

specifies the ODS output object.

Options

`n`

specifies the relative line that contains the footnote.

Range:  1–10

Tip:  The OBFOOTN line with the highest number appears on the bottom line. If you omit `n`, SAS assumes a value of 1. Therefore, you can specify OBFOOTN or OBFOOTN1 for the first text line.

Tip:  You can create footnotes that contain blank lines between them. For example, if you specify text with an OBFOOTN statement that is followed by an OBFOOTN3 statement, then a blank line separates the two lines of text.

`text`

specifies the text string.

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, you can embed the items at the position where you want them to appear. For more information, see “The DOCUMENT PROCEDURE and BY-Groups” on page 233.

Requirement:  All text strings must be enclosed by quotation marks.
Caution:  If you use the OBFOOTN statement without a text string, then all existing footnotes for the specified output object are deleted.

OBPAGE Statement

Creates or deletes a page break for an output object

Featured In:  Example 3 on page 253
OBPAGE output-object </DELETE AFTER>;

Required Arguments

output-object specifies the name of the output object.

Without Options

If no options are specified, then the OBPAGE statement inserts a page break before an output object.

Options

AFTER inserts a page break after an output object.

Tip: To delete a page break after an output object, you must use the AFTER option as well as the DELETE option.

DELETE removes the page break for an output object.

OBSTITLE Statement

Creates or modifies subtitles

Featured In: Example 3 on page 253

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

Required Arguments

output-object specifies the ODS output object.

Options

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

LEFT aligns the text string with the left margin.
CENTER
  aligns the text string in the center between the left and right margins.

RIGHT
  aligns the text string with the right margin.

\( n \)
  specifies the relative line that contains the subtitle.

**Range:** 1–10

**Tip:** The OBSTITLE line with the highest number appears on the bottom line. If you omit \( n \), SAS assumes a value of 1. Therefore, you can specify OBSTITLE or OBSTITLE1 for the first text line.

**Tip:** You can create subtitles that contain blank lines between them. For example, if you specify text with an OBSTITLE statement that is followed by an OBSTITLE3 statement, then a blank line separates the two lines of text.

\text
  specifies the text string.

  You can customize subtitles by inserting BY variable values (#BYVALn), BY variable names (#BYVARN), or BY lines (#BYLINE) into subtitles that are specified in PROC DOCUMENT steps. After you specify text, you can embed the items at the position where you want them to appear. For more information, see “The DOCUMENT PROCEDURE and BY-Groups” on page 233.

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

**Caution:** If no arguments are specified, then the OBSTITLE statement deletes all existing subtitles for the specified output object only.

---

**OBTITLE Statement**

Creates or modifies title lines for the output

**Tip:** The OBTITLE is similar to the global TITLE statement.

**Featured In:** Example 3 on page 253

\begin{verbatim}
OBTITLE<n> output-object <'text'>;
\end{verbatim}

**Required Arguments**

\textbf{output-object}
  specifies the name of the output object.

**Options**

\( n \)
  specifies the relative line that contains the title.

**Range:** 1–10
Tip: The OBTITLE line with the highest number appears on the bottom line. If you omit \( n \), SAS assumes a value of 1. Therefore, you can specify OBTITLE or OBTITLE1 for the first text line.

Tip: You can create titles that contain blank lines between them. For example, if you specify text with an OBTITLE statement that is followed by an OBTITLE3 statement, then a blank line separates the two lines of text.

text

specifies the text string.

You can customize titles by inserting BY variable values (#BYVAL\( n \)), BY variable names (#BYVAR\( n \)), or BY lines (#BYLINE) into output titles that are specified in PROC DOCUMENT steps. After you specify the text, you can embed the items at the position where you want them to appear. For more information, see “The DOCUMENT PROCEDURE and BY-Groups” on page 233.

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 Statement**

Assigns a different name to a file location or output object

RENAME path-1 TO path-2;

**Required Arguments**

*path-1*

specifies the current file location or output object.

*path-2*

specifies the new name of the file location or output object.

---

**REPLAY Statement**

Displays one or more entries to the specified open ODS destination(s)

Default: If you omit the LEVELS= option, then all levels of the file are displayed to all open destinations.

Featured In: Example 2 on page 248 and Example 3 on page 253

REPLAY <path <, path-2, ...path-n>> </<LEVELS= value | ALL> <DEST= (ODS-destination(s)))>>;
Options

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

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

DEST= (ODS-destination(s))
specifies one or more ODS destinations where you want your output objects to be displayed.
Requirement: When you specify the DEST= option, you must surround the ODS destinations with parentheses and separate each destination with a blank space. For example, DEST=(HTML RTF LISTING)
Tip: When you specify only one destination, you do not need to use parentheses. For example, DEST=HTML
See Also: For information about ODS destinations, see SAS Output Delivery System: User's Guide.

LEVELS= ALL | value
specifies the depth of the path.
ALL
specifies that all levels of the path are displayed to all open destinations.
value
specifies the numeric value of the level.

path
specifies the location of an entry. An entry can be one or more file locations, links, or output objects.
Requirement: When you specify more than one path, separate the paths with a comma.

---

SETLABEL Statement

Assigns a label to the specified path

SETLABEL path 'label';

Required Arguments

label
specifies the text of the label. You can customize labels by inserting BY variable values (#BYVAL), BY variable names (#BYVAR), or BY lines (#BYLINE) into labels
that are specified in PROC DOCUMENT steps. For more information, see “The DOCUMENT PROCEDURE and BY-Groups” on page 233.

**Requirement:** The label must be enclosed in quotation marks.

**path**

specifies the location of a link, output object, or file location.

---

**UNHIDE Statement**

Enables the output of a hidden entry to be displayed when it is replayed

**UNHIDE path <, path-2, ...path-n>**;

**Required Arguments**

**path**

specifies the location of a link, output object, or file.

**Requirement:** When you specify more than one path, separate the paths with a comma.

---

**The DOCUMENT PROCEDURE and BY-Groups**

You can customize labels, titles, and footnotes with the following statements by inserting BY variable values (#BYVAL), BY variable names (#BYVAR), or BY lines (#BYLINE) in labels that are specified in PROC DOCUMENT steps:

- OBANOTE statement
- OBBNOTE statement
- OBFOOTN statement
- OBSTITLE statement
- OBTITLE statement
- SETLABEL statement

**Note:** The #BYVAL, #BYVAR, and #BYLINE substitutions will only show up for replayed output objects that belong to a BY group. Examples of output objects that do not belong to a BY group are:

- data sets that are imported into a document with IMPORT statement
- notes 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 the following form:

#BYVALn | #BYVAL(variable-name)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value in the label.

Follow these rules when you use #BYVAL in a valid statement of a PROC DOCUMENT step:

- Specify the variable that is used by #BYVAL in the BY statement.
- Insert #BYVAL in the specified text string at the position where you want the substitution text to appear.
- Follow #BYVAL with a delimiting character, either a space or other nonalphanumeric character (for example, a quotation mark) that ends the text string.
- If you want the #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables).

Specify the variable with one of the following:

n
specifies which variable in the BY statement that #BYVAL should use. The value of n indicates the position of the variable in the BY statement.

Example: #BYVAL2 specifies the second variable in the BY statement.

variable-name
names the BY variable.

Example: #BYVAL(YEAR) specifies the BY variable, YEAR.

Tip: Variable-name is not case sensitive.

Requirement: You must enclose variable-name in parentheses.

#BYVARn | #BYVAR(variable-name)
substitutes the name of the BY variable or label that is associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label.

Follow these rules when you use #BYVAR in a valid statement of a PROC DOCUMENT step:

- Specify the variable that is used by #BYVAR in the BY statement.
- Insert #BYVAR in the specified text string at the position where you want the substitution text to appear.
- Follow #BYVAR with a delimiting character, either a space or other nonalphanumeric character (for example, a quotation mark) that ends the text string.
- If you want the #BYVAR substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables).
- Specify the variable with one of the following:

n
specifies which variable in the BY statement that #BYVAR should use. The value of n indicates the position of the variable in the BY statement.

Example: #BYVAR2 specifies the second variable in the BY statement.

variable-name
names the BY variable.

Example: #BYVAR(SITES) specifies the BY variable SITES.
Tip: Variable-name is not case sensitive.

Requirement: You must enclose variable-name in parentheses.

#BYLINE
substitutes the entire BY line without leading or trailing blanks for #BYLINE in the text string and displays the BY line in the label.

Concepts: DOCUMENT Procedure

What Is an ODS Document?

Definition
An ODS document is a hierarchical file of output objects that is created from a procedure or data query. The hierarchy is controlled by the internal logic of the procedure or data query.

What Does an ODS Document Include?
In an ODS document, each level of the hierarchical file represents a path which refers to the location of a file, link, or output object. An output object can be a
- table
- graph
- equation
- note.

What Is Not Included in an ODS Document?
An ODS document does not store
- SAS logs
- SAS system options
- procedure options
- ODS options
- SAS/GRAPH options
- SAS/GRAPH external graph titles
- GRSEGs (references to GRSEGs, but not GRSEGs themselves, are stored.)

ODS Document Persistence
An ODS document is a member of a SAS library. Therefore, you can browse, edit, and replay the output contained in the ODS document to any ODS destination without rerunning your SAS programs that created the initial output. An ODS document persists in the SAS System until the document, or the SAS library containing the document, is deleted. Thus an ODS document that was created in the SASUSER library, or in another permanent SAS library, can persist indefinitely because it is considered a
permanent archive of SAS procedure output. However, an ODS document that is created in the WORK library does not persist longer than the SAS session that created it. For information about SAS data libraries, see *SAS Language Reference: Concepts*

What Is an ODS Document Path?

Definition of ODS Document Path

Because an ODS document is stored as an item store, this file format enables client applications to define a “hierarchal file system within a file.” This is similar to a directory system in a Windows operating environment, or a partitioned data set in a mainframe operating environment. Therefore, an ODS document path means the location of an entry.

Entry Names

Entry names
- must be alphanumeric
- must begin with an alphabetical character
- can contain underscores
- can have no more than 32 characters
- are preserved with casing (uppercase, lowercase, or mixed case) that is specified in the mainframe operating environment
- can have labels which are no more than 256 characters.

Entries are inserted into an ODS document in the following three ways:
- ordered by insertion, which is the default order
- ordered by ascending date-time stamp
- ordered alphabetically.

Understanding Sequence Numbers

Entry names are not required to be unique within an ODS document. However, they are uniquely identifiable because they contain sequence numbers. Every entry in an ODS document, except for the root file location, has a sequence number. A sequence number is a positive integer that is unique with respect to the name of the entry within the same file location level. Entries are assigned sequence numbers according to the sequence in which they are added to a file location. For example, the first entry `myname` is assigned a sequence number 1, `myname#1`. The second entry `myname` is assigned a sequence number 2, `myname#2`. Sequence numbers are never reassigned, unless all entries with the same name are deleted. In this case, the sequence numbers are reset to an initial number of 1.

ODS Documents and Base SAS Procedures

You can create an ODS document from almost any Base SAS procedure. The FREQ, PRINT, REPORT, and TABULATE procedures use table definitions that are created by the user, and not defined by a template in ODS. These procedures use custom table definitions, custom data components, and custom formats for their output objects.
Nevertheless, the ODS document and all of its features are supported for the TABULATE procedure. Except for the crosstabs tables in the FREQ procedure, the remaining output objects are supported in an ODS document. ODS documents support some features of PROC PRINT. For example, BY-group processing is not supported, and the REPORT procedure is not supported.

**Getting Familiar with Output Objects**

An output object can be one of the following:

- table
- note
- equation.

Output objects have associated information and attributes. Some or all of these attributes pertain to output objects.

- **after-note** is the note assigned to the output object by the procedure that produced the object. This note is displayed every time the output object is displayed. After-notes display after the output object.

- **before-note** is the note assigned to the output object by the procedure that produced the object. This note is displayed every time the output object is displayed. Before-notes display before the output object.

- **footnote** is created by the FOOTNOTE statement and is displayed when the output object is created.

- **page break** causes a page break prior to displaying the output object and any associated titles and notes.

- **subtitle** is the title that is assigned to the output object by the procedure that produced the output object. This title is displayed every time a new page of output is started.

- **title** is created by the TITLE statement and is displayed when the output object is created.

Here is the order in which the attributes of an output object are displayed:

1. page break
2. titles
3. subtitles
4. before-notes
5. output object
6. after-notes
7. footnotes

**How Do ODS Documents Interact across Operating Environments?**

**Compatibility across SAS Versions**

An ODS document that is created in the current version of SAS is compatible with later versions of SAS. In most cases, an ODS document created in a later version of SAS will still be compatible with today's 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.

### ODS Documents in the Documents Window

#### Why Use the Documents Window?

The Documents window displays your ODS documents in a hierarchical tree structure. You can use the Documents window to do the following:

- view all of your ODS documents including ODS documents stored in SAS libraries
- organize, manage, and customize the layout of the entries contained in your ODS documents
- view the property information of your ODS documents
- replay entries
- rename, copy, move, or delete your ODS documents
- create shortcuts to your ODS documents.

For a comparison of the Documents window to the Results Window, see “Comparisons between the Documents Window and the Results Window” on page 242

#### Viewing an ODS Document in the Documents Window

To view the Documents window, submit the following command in the command bar:

```
odsdocuments
```

The following display shows the Documents window that contains the ODS document named `Sasuser.Univ`. In the display, you can see that `Sasuser.Univ` contains several file location levels. The `Exponential_x` file location contains the `Exp` output object. When you double-click an output object, such as `Exp`, that output object is replayed in the Results Viewer window to all open destinations.
A Documents window contains the following items:

- **entry**: is an output object, link, or file location.
  
  \[\text{Note: Only output objects of the type document are displayed in the Documents window.}\]

- **file location**: is a grouping of ODS document entries.

- **link**: is a symbolic link from one specified output object to another output object.
  
  \[\text{Note: Within the Documents window, a link is called a shortcut.}\]

- **ODS document**: is the name of your ODS document.

**ODS Document Icon**

The Results window and the Documents window use the following icon to indicate an ODS document output object:

**Display 6.2  ODS Document Icon**

---

*Operating Environment Information*: The ODS Documents window on z/OS has the same functionality, but does not use graphical icons.
Using the Documents Window Pop-up Menu

The Documents window has a pop-up menu with features that are also available through batch processing. To view the Documents window pop-up menu, follow these steps:

1. Type `odsdocuments` in the command bar. The Documents window appears.
2. Right-click any entry in the Documents window. The pop-up menu appears.

**Display 6.3  Pop-up Menu for the Documents Window**

![Pop-up Menu for the Documents Window](image)

The following table describes the pop-up menu item features. The availability of each pop-up menu item depends on which entry you select in the Documents window.

**Table 6.1  Tasks You Can Do with the Documents Window Pop-up Menu**

<table>
<thead>
<tr>
<th>Task</th>
<th>Menu item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the selected object in the Results Viewer</td>
<td>Open</td>
</tr>
<tr>
<td>Select a new ODS destination output type</td>
<td>Open As</td>
</tr>
<tr>
<td>Open a window in tree view and list view</td>
<td>Explore From Here</td>
</tr>
<tr>
<td>Create a new folder</td>
<td>New Folder</td>
</tr>
<tr>
<td>Remove the selected entry from the Documents window</td>
<td>Cut</td>
</tr>
<tr>
<td>Copy the selected entry to system memory</td>
<td>Copy</td>
</tr>
<tr>
<td>Paste the copied entry to the selected location</td>
<td>Paste</td>
</tr>
<tr>
<td>Create a shortcut to the entry</td>
<td>Create Shortcut</td>
</tr>
<tr>
<td>Delete the selected entry</td>
<td>Delete</td>
</tr>
<tr>
<td>Rename the selected entry</td>
<td>Rename</td>
</tr>
<tr>
<td>Show the entries that were previously excluded</td>
<td>Show Excluded</td>
</tr>
</tbody>
</table>
The DOCUMENT Procedure

<table>
<thead>
<tr>
<th>Task</th>
<th>Menu item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove from the tree, but do not delete the</td>
<td>Exclude</td>
</tr>
<tr>
<td>selected entry</td>
<td></td>
</tr>
<tr>
<td>Expand all the levels of the tree</td>
<td>Expand All</td>
</tr>
<tr>
<td>Collapse all the levels in the tree</td>
<td>Collapse All</td>
</tr>
<tr>
<td>Replay the selected entry to all open ODS</td>
<td>Replay</td>
</tr>
<tr>
<td>destinations</td>
<td></td>
</tr>
<tr>
<td>Print the selected entry</td>
<td>Print</td>
</tr>
<tr>
<td>Display the properties of the selected entry</td>
<td>Properties</td>
</tr>
</tbody>
</table>

* Available menu choices vary, depending on the selected entry.

## ODS Documents in the Results Window

### Why 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. You can use the Results window to do the following:

- view the output object types that are created when you run a SAS program in your current SAS session. SAS creates an output object for each ODS destination that was open at the time you executed a procedure during your current SAS session only.
- view the results after you create a new output object from the Documents window using the Open As or Replay feature.
- view the properties of an entry.
- delete or rename entries.

See “Comparisons between the Documents Window and the Results Window” on page 242.

### Viewing Entries in the Results Window

To view the Results window, submit the following command in the command bar:

```sas
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.
Comparisons between the Documents Window and the Results Window

The following table shows you the tasks that you can and cannot do in the Documents window and in 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 your output objects</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>View the property information of your 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 your SAS documents</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Viewing the Properties of an Entry

Any entry that you select either in the Results window or in the Documents window has an associated Properties window. To view the properties of an entry, follow these steps:

1. Select an entry either from the Results Window or from the Documents window.
2. Right-click the entry. A pop-up menu appears.

Display 6.5  Entry Properties Window

Items will vary, depending on the entry that you select in the Documents or Results windows. The Properties window for an ODS document output object can contain the following items:

Created is the date that the entry was created.

Document is the SAS filename where the entry is located. The filename is in the form of libref.filename

Document path 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 will change.

Modified is the date that the entry was modified.

Name is the name of the entry.

Path is the storage location inside the document of the entry.

Type is the classification of the entry.
Creating Shortcuts in the Documents Window

The Documents window pop-up menu provides you with a **Create Shortcut** option. Shortcut links are useful when you are creating output that uses the same entry in more than one place. Instead of copying the entry to each location, consider using a shortcut. Shortcuts have the following advantages:

- Because a shortcut is a link to the original entry, any changes that you make to the original entry will appear when you select the shortcut.
- A shortcut uses fewer computer resources.

To create a shortcut, do the following:

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.

Examples: DOCUMENT Procedure

Example 1: Navigating the File Location and Listing the Entries

**Procedure features:**
- ODS DOCUMENT statement options:
  - NAME=
- DOC statement option:
  - NAME=
- LIST statement options:
  - entry
    - LEVELS=
    - DETAILS
- DIR statement option:
  - path

**ODS Destinations:**
- DOCUMENT
- LISTING
- HTML

**Procedure output:**
- PROC DOCUMENT

This example shows you how to do the following:
- name an ODS document
- see what ODS documents exist
- open a document for browsing or editing purposes
- list one or more entries
- change file locations.
Program

Set the SAS system options. The NODEATE option suppresses the display of the date and time in the output. The NONUMBER option suppresses the printing of page numbers.

```sas
options nodate nonumber;
```

Create the DISTRDATA data set. The DISTRDATA data set contains the statistical information that PROC UNIVARIATE uses to create the histograms.

```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;
```

Create the ODS document UNIV and open the DOCUMENT destination. The ODS DOCUMENT statement opens the document destination. The NAME= option assigns the name UNIV to the ODS document that contains the information from this PROC UNIVARIATE program. Note that by default UNIV will be created in the WORK library. You must assign a libref if you want UNIV to be created in a permanent library.

```sas
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.

```sas
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.

```sas
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 you do not close the DOCUMENT destination, you will be unable to see DOCUMENT procedure output.
ods document close;
title;

View your documents, choose a document, and list the entries of the document you open. The DOC statement (with no arguments specified) prints a listing of all of the available documents that are in the SAS System (see ). The DOC statement with the NAME= option specifies the current document, WORK.UNIV. The LIST statement with the LEVELS=ALL option lists detailed information on all levels of the document WORK.UNIV (see ).

```sas
proc document;
  doc;
    doc name=univ;
    list/levels=all;
run;
```

Set the path to EXPONENTIAL, list the contents of the EXPONENTIAL file location, select a table, and list the details of the table you selected. The DIR statement changes your current file location to \texttt{univariate\#2/exponential\_x/fitteddistributions/exponential}. The path \texttt{univariate\#2/exponential\_x/fitteddistributions/exponential} was obtained from the listing of the WORK.UNIV document (see Display 1.6). The LIST statement (with no arguments) lists the contents of EXPONENTIAL (see List of the EXPONENTIAL#1 Entry). The LIST fitquantiles/details statement specifies that ODS opens the FitQuantiles table and lists its details (see Details of the FitQuantiles#1 Table).

```sas
dir univariate\#2/exponential\_x/fitteddistributions/exponential;
  list;
    list fitquantiles/details;
run;
```

Terminate the DOCUMENT procedure. You must specify the QUIT statement to terminate the DOCUMENT procedure. If you do not specify QUIT, then you will not be able to view DOCUMENT procedure output.

```sas
quit;
```
Output

Display 6.6 List of ODS Documents

The following display shows that there are currently two ODS documents, SASUSER.ODSGLM and WORK.UNIV.

Display 6.7 List of the Contents of WORK.UNIV

The following display shows the entries of the ODS document WORK.UNIV and the properties of those entries.
Example 2: Opening and Listing ODS Documents

Procedure features:
- PROC DOCUMENT statement option:
  NAME=
- DIR statement
- LIST statement option:
  DETAILS
  LEVELS
- REPLAY statement

ODS Destinations:
- DOCUMENT
LISTING
PDF

Procedure output:
PROC DOCUMENT
PROC UNIVARIATE
DATA SET: DISTRDATA on page 245

This example shows you how to do these tasks:
- open an ODS document
- replay a table and send the output to the LISTING and PDF destinations
- list the entries in an ODS document
- change file locations
- list the details of a specified entry
- replay an ODS document to a PDF file.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The NONUMBER option suppresses the printing of page numbers.
```
options nodate nonumber;
```

Open the ODS document WORK.UNIV. The PROC DOCUMENT statement with the NAME= option specified, opens the ODS document WORK.UNIV for updates.
```
proc document name=univ;
```

Specify that you want to replay your output to a PDF file. The ODS PDF statement opens the PRINTER destination and replays the histogram to the PDF destination. The FILE= statement sends all output objects to the external file that you specify.
```
ods pdf file= "your_file.pdf";
```

List the entries that are associated with the current document and replay a histogram. The LIST statement with the LEVELS=ALL option specified, lists detailed information on all levels of the current document WORK.UNIV (see Display 6.7 on page 247). The REPLAY statement replays the NORMAL#1 entry to all open ODS destinations (see Display 6.11 on page 251).
```
list/levels=all;
replay univariate#1\Normal_x#1\Normal#1;
```

View the file EXPONENTIAL, list the details of the FitQuantiles table, and replay the FitQuantiles table. The DIR statement changes the current file location to univariate#2\exponential_x\fitteddistributions\exponential#1. The LIST statement (with no arguments) lists the entries in the EXPONENTIAL file location (see Display 6.12 on page 251).

The LIST statement with the DETAILS option specifies the listing of the properties of the entry FitQuantiles table (see ). The REPLAY statement replays FITQUANTILES to the PDF destination.


```
dir univariate\2\exponential_x\fitteddistributions\exponential\1;
list;
list fitquantiles/details;
replay fitquantiles;
run;
```

**Terminate the DOCUMENT procedure and close the PDF destination.** You must specify the QUIT statement to terminate the DOCUMENT procedure. If you do not specify QUIT, then you will not be able to view DOCUMENT procedure output. The ODS PDF CLOSE statement closes the PDF destination and all the files that are associated with it. If you do not close the destination, then you will not be able to view the files.

```
quit;
ods pdf close;
```

**Output**

**Display 6.10** List of the Contents of WORK.UNIV

The following display shows the contents of WORK.UNIV.

![Output](attachment:image)

```
Display 6.11  Replayed Normal Distribution Histogram

Display 6.12  List of the EXPONENTIAL#1 File Location
**Display 6.13**  Details of the FitQuantiles#1 Table

The following display shows the properties of the FitQuantiles#1 table, viewed in the SAS Output window.

**Display 6.14**  Replayed FitQuantiles#1 Table

The following display shows the replayed FitQuantiles#1 table that was sent to the LISTING destination.
Example 3: Managing Entries

Procedure features:
PROC DOCUMENT statement option:
   NAME=
DIR statement
LIST statement option:
   LEVELS=
NOTE statement
OBANOTE statement
OBBNOTE statement
OBFOOTN statement
OBPAGE statement
OBSTITLE statement
OBTITLE statement
REPLAY statement

ODS Destinations:
   DOCUMENT
   HTML
   LISTING

Procedure output:
   PROC CONTENTS

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 a before-note and an after-note to the output
- change the subtitle of the output

The following display is page 1 of the ODS document WORK.UNIV that was sent to the PDF destination. You can browse the output by clicking the entries.
Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The PAGENO= option specifies the starting page number.

```sas
options nodate pageno=1;
```

Close the LISTING destination and open the DOCUMENT destination. The NAME= option creates an ODS document named CLASS.

```sas
ods listing close;
ods document name=class;
```

Specify a global title and footnote. The TITLE statement creates a title that is used until you change it with another statement. The FOOTNOTE statement creates a footnote that is used until you change it with another statement. For information about the global FOOTNOTE and TITLE statements, see SAS Language Reference: Dictionary.

```sas
title 'Title Specified with the Global TITLE Statement';
footnote 'Footnote Specified with the Global FOOTNOTE Statement';
```

View the contents of the SAS data set. The CONTENTS procedure shows the contents of a SAS data set sashelp.class.

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

Close the DOCUMENT destination. The entries in the ODS document CLASS are used in the remainder of this example.

```sas
ods document close;
```

Create LISTING and HTML output. The ODS LISTING statement opens the LISTING destination and creates listing output.

The ODS HTML statement opens the HTML destination and creates HTML 4.0 output. The STYLE= option specifies that ODS use the style definition D3D.

```sas
ods listing;
ods html file='your_file.html' style=d3d;
```
Change the global title. The OBTITLE statement assigns a new title to the Attributes\#1 object. See Title, Subtitle, and Before-note on page 257.

- The NAME= option specifies the current ODS document.
- The LIST statement with the LEVELS=ALL option shows a list of entries in the CLASS document. Note that PROC DOCUMENT is still running after the RUN statement executes.
- The DIR statement changes the current path to \\Contents\#1\DataSet\#1.
- REPLAY generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
   list /levels=all;
run;
   dir \Contents\#1\DataSet\#1;
run;
   obtitle Attributes\#1 'Title Specified with the OBTITLE Statement';
    replay;
run;
quit;
```

Add a before-note to the output. The OBBNOTE statement assigns a before-note to the Attributes\#1 object. See Title, Subtitle, and Before-note on page 257.

- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to \\Contents\#1\DataSet\#1.
- The REPLAY statement generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
   dir \Contents\#1\DataSet\#1;
run;
   obbnote Attributes\#1 'Add Before Note using the OBBNOTE Statement';
    replay;
run;
quit;
```

Change the global footnote. The OBFOOTN statement assigns a new footnote to the object Variables\#1.

- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to \\Contents\#1\DataSet\#1.
- The REPLAY statement generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
   dir \Contents\#1\DataSet\#1;
run;
   obfootn Variables\#1 'Change Footnote using the OBFOOTN Statement';
    replay;
run;
quit;
```
Add an after-note. The OBANOTE statement assigns an after-note to the Attributes object. See Display 6.18 on page 258.

- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to \Contents\DataSet.
- The REPLAY statement generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
  dir \Contents\DataSet;
  run;
  obanote Attributes 'After Note Specified by the OBANOTE Statement';
  replay;
run;
quit;
```

Change the subtitle of the output. The OBSTITLE statement changes the subtitle. The subtitle identifies the procedure that produced the output. See Title, Subtitle, and Before-note on page 257.

- The NAME= option specifies the current ODS document.
- The DIR statement changes the current file location to \Contents\DataSet.
- The REPLAY statement generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.

```plaintext
proc document name=class;
  dir \Contents\DataSet;
  run;
  obstitle Attributes 'Subtitle Specified with the OBSTITLE Statement';
  replay;
run;
quit;
```

Add a note to the document. The NOTE statement adds a note object named ADDNOTE to the ODS document. See Note Is Added and Footnote Is Changed on page 258.

- 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';
  list /levels=all;
run;
quit;
```

Add a page break to the output. The OBPAGE statement inserts a page break.

- The NAME= option specifies the current ODS document.
- The REPLAY statement generates output for all open ODS destinations.
- The QUIT statement terminates PROC DOCUMENT.
proc document name=class;
    obpage \Contents#1\DataSet#1\Variables#1;
    replay;
run;
quit;

Close the HTML and LISTING destinations. The ODS _ALL_ CLOSE statement closes all open ODS output destinations so that you can view the output.

ods _all_ close;

Output

Display 6.16  Global Title and Footnote

Display 6.17  Title, Subtitle, and Before-note
Display 6.18  After-note Is Specified by the OBANOTE Statement

Display 6.19  Note Is Added
The TEMPLATE Procedure

Chapter 7............ TEMPLATE Procedure: Overview 261
Chapter 8............ TEMPLATE Procedure: Managing Template Stores 271
Chapter 9............ TEMPLATE Procedure: Creating a Style Definition 285
Chapter 10........... TEMPLATE Procedure: Creating Tabular Output 367
Chapter 11........... TEMPLATE Procedure: Creating Markup Language Tagsets 551
Introduction

Why Use 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 definitions for various types of output:

- styles
- tables
- columns
- headers
- footers
- tagsets

ODS then uses these definitions to produce formatted output.

You can also use the TEMPLATE procedure to navigate and manage the definitions stored in templates stores. Here are some tasks that you can do with PROC TEMPLATE:

- edit an existing definition
- create links to an existing definition
- change the location where you write new definitions
- search for existing definitions
- view the source code of a definition
What Can You Do with the TEMPLATE Procedure?

Modify a Table Definition that a SAS Procedure Uses

The following output shows the use of a customized table definition for the Moments output object from PROC UNIVARIATE. The program used to create the modified table definition

- creates and edits a copy of the default table definition.
- edits a header within the table definition.
- sets column attributes to enhance the appearance of both the HTML and the Listing output.

Output 7.1  Listing Output (Customized Moments Table) from PROC UNIVARIATE

<table>
<thead>
<tr>
<th>Custom Moments Table</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The UNIVARIATE Procedure</td>
<td></td>
</tr>
<tr>
<td>Variable: CityPop_90 (1990 metropolitan pop in millions)</td>
<td></td>
</tr>
<tr>
<td>Moments</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>51</td>
</tr>
<tr>
<td>Mean</td>
<td>3.87701961</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>5.16465302</td>
</tr>
<tr>
<td>Skewness</td>
<td>2.87109259</td>
</tr>
<tr>
<td>Uncorrected SS</td>
<td>2100.27737</td>
</tr>
<tr>
<td>Coeff Variation</td>
<td>133.21194</td>
</tr>
</tbody>
</table>
Display 7.1  Customized HTML Output (Customized Moments Table) from PROC UNIVARIATE (Viewed with Microsoft Internet Explorer)

<table>
<thead>
<tr>
<th>Moments</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>51</td>
<td>Sum Weights</td>
</tr>
<tr>
<td>Mean</td>
<td>3.87701961</td>
<td>Sum Observations</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>5.16485302</td>
<td>Variance</td>
</tr>
<tr>
<td>Skewness</td>
<td>2.87109259</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>Uncorrected SS</td>
<td>2100.27737</td>
<td>Corrected SS</td>
</tr>
<tr>
<td>Coeff Variation</td>
<td>133.21194</td>
<td>Std Error Mean</td>
</tr>
</tbody>
</table>

**Custom Moments Table**

The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

**Modify a Style Definition**

When you are working with style definitions, you are more likely to modify a style definition that SAS supplies than to write a completely new style definition. The output below uses the Styles.Default definition that SAS provides, but includes changes made to the style definition in order to customize the output’s appearance. The Display 7.2 on page 264 shows changes made to both the contents file and the body file in the HTML output. In the contents file, the modified style definition 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 definition 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 headers. 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.
Create Your Own Tagset

Tagsets are used to create custom markup. You can create your own tagsets, extend existing tagsets, or modify a tagset definition that SAS supplies. The following display shows the results from a new tagset \texttt{TAGSET.MYTAGS}.
Display 7.3  MYTAGS.CHTML Output (Viewed with Microsoft Internet Explorer)

To see the customized CHTML tagset, view the source from your web browser:

- Select from your browser's tool bar:

```
View ▶ Source
```

These are my new colspecs

<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>
Terminology: TEMPLATE Procedure

The following 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.

item store is a member of a SAS data 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 data library that the libref references. For example, the SAS registry is stored in two items stores, SASUSER.REGISTRY and SASHELP.REGISTRY.

template store is an item store which stores definitions that were created by the TEMPLATE procedure. Definitions that SAS provides are in the item store SASHELP.TMPLMST. You can store definitions that you create in any template store 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 data library that the libref references.

style definition describes how to display the presentation aspects (color, font face, font size, and so on) of your SAS output. A style definition determines the overall appearance of the documents that use it. Each style definition is composed of style elements. Style definitions do not apply to the LISTING destination, which produces plain text output.

style element is a collection of style attributes that apply to a particular part of the output. For example, a style element may contain instructions for the presentation of column headers or for the presentation of the data inside cells. Style elements may also specify default colors and fonts for output that uses the style definition. 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 FONT_STYLE= attribute specifies whether to use a Roman, a slant, or an italic font.

table definition describes how to display the output for a tabular output object. (Most ODS output is tabular.) A table definition 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 definition 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 definition. Any table definition can then reference these table elements. For more information about defining columns, headers, and footers outside of the table definition, see Chapter 10, “TEMPLATE Procedure: Creating Tabular Output,” on page 367.

tagset definition

specifies instructions for creating a markup language for your SAS output. The resulting output contains embedded instructions in order to define layout and some content. Each tagset definition contains event definitions and event attributes that control the generation of the output. SAS provides tagset definitions for a variety of markup languages. With the TEMPLATE procedure, you can modify any of these SAS tagsets, or you can create your own tagsets.

event

specifies the text that the markup destination produces when the specified event occurs. For example, the definition of an event called ROW might specify to place the appropriate tags for starting a row at the beginning of an event and the appropriate tags for ending a row at the end of the event. SAS procedures that generate ODS output use a standard set of events, which you can customize with the TEMPLATE procedure.

### PROC TEMPLATE Statements by Category

The following table lists and describes the categories and statements used in the TEMPLATE procedure.

<table>
<thead>
<tr>
<th>Task</th>
<th>Statements Category</th>
<th>Statements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate template stores and manage ODS definitions</td>
<td>Template store</td>
<td>DELETE</td>
<td>Deletes the specified definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINK</td>
<td>Creates a link to an existing definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIST</td>
<td>Lists items in one or more template stores</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PATH</td>
<td>Specifies the locations to write to or read from when creating or using PROC TEMPLATE definitions, and the order in which to search for them</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOURCE</td>
<td>Writes the source code for the specified definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TEST</td>
<td>Tests the most recently created definition by binding it to the specified data set</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>Create or modify ODS style definitions</td>
<td>Style</td>
<td>DEFINE STYLE</td>
<td>Creates a style definition for any destination that supports the STYLE= option</td>
</tr>
<tr>
<td>Create and modify ODS table, column, header, and footer definitions</td>
<td>Tabular</td>
<td>DEFINE COLUMN DEFINE FOOTER DEFINE HEADER DEFINE TABLE</td>
<td>Edits an existing definition</td>
</tr>
<tr>
<td>Create or modify markup language tagsets</td>
<td>Markup language tagsets</td>
<td>DEFINE TAGSET</td>
<td>Creates a definition for a tagset</td>
</tr>
</tbody>
</table>

**Syntax: TEMPLATE Procedure**

```plaintext
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 definition-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;

DELETE definition-path </ STORE=libref.template-store >;

EDIT definition-path-1 <AS definition-path-2> </ STORE=libref.template-store > ;
statements-and-attributes
END;
LINK definition-path-1 TO definition-path-2 </ option(s) >;
LIST <starting-path> </ option(s) >;
PATH location(s);
SOURCE definition-path </ option(s) >;
TEST DATA= SAS-data-set </ STORE=libref.template-store >;

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a column definition</td>
<td>“DEFINE COLUMN Statement” on page 374</td>
</tr>
<tr>
<td>Create a footer definition</td>
<td>“DEFINE FOOTER Statement” on page 395</td>
</tr>
<tr>
<td>Create a header definition</td>
<td>“DEFINE HEADER Statement” on page 395</td>
</tr>
<tr>
<td>Create a style definition</td>
<td>“DEFINE STYLE Statement” on page 288</td>
</tr>
<tr>
<td>Create a table definition</td>
<td>“DEFINE TABLE Statement” on page 410</td>
</tr>
<tr>
<td>Create a tagset definition</td>
<td>“DEFINE TAGSET Statement” on page 552</td>
</tr>
<tr>
<td>Delete the specified definition</td>
<td>“DELETE Statement” on page 273</td>
</tr>
<tr>
<td>Edit an existing definition</td>
<td>“EDIT Statement” on page 373</td>
</tr>
<tr>
<td>Create a link to an existing definition</td>
<td>“LINK Statement” on page 273</td>
</tr>
<tr>
<td>List items in one or more template stores</td>
<td>“LIST Statement” on page 274</td>
</tr>
<tr>
<td>Specify the locations to write to or read from when creating or using PROC TEMPLATE definitions, and the order in which to search for them</td>
<td>“PATH Statement” on page 276</td>
</tr>
<tr>
<td>Write the source code for the specified definition to the SAS log</td>
<td>“SOURCE Statement” on page 277</td>
</tr>
<tr>
<td>Test the most recently created definition by binding it to the specified data set</td>
<td>“TEST Statement” on page 279</td>
</tr>
</tbody>
</table>

Where to Go from Here

- Managing the various definitions stored in template stores: For reference information about the PROC TEMPLATE statements that help you manage and navigate around the many ODS definitions, see Chapter 8, “TEMPLATE Procedure: Managing Template Stores,” on page 271.
- **Modifying an existing style definition or creating your own style definition**: For reference information about the style definition statements in PROC TEMPLATE, see Chapter 9, “TEMPLATE Procedure: Creating a Style Definition,” on page 285.

- **Creating and modifying ODS tabular output**: For reference information about the tabular definition statements in PROC TEMPLATE, see Chapter 10, “TEMPLATE Procedure: Creating Tabular Output,” on page 367.

- **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 11, “TEMPLATE Procedure: Creating Markup Language Tagsets,” on page 551.
Overview: Template Stores

What Is a Template Store?

A template store is an item store which stores definitions that were created by the TEMPLATE procedure. Definitions that SAS provides are in the item store SASHELP.TMPLMST. You can store definitions that you create in any template store 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 data library that the libref references.

Why Use the TEMPLATE Procedure to Manage Template Stores?

You can use the TEMPLATE procedure to manage and navigate the template stores that store the definitions that SAS supplies or that you create. The TEMPLATE procedure enables you to manage the template stores by

- deleting column, header, footer, style, table, or tagset definitions
To navigate your way around the template stores you can
- create links to existing definitions
- specify which locations to write to or read from when you create or use PROC TEMPLATE definitions, and specify the order in which to search for them.

---

**Terminology**

For definitions of terms used in this section, see “Terminology: TEMPLATE Procedure” on page 266.

---

**Template Store Syntax: TEMPLATE Procedure**

```sas
PROC TEMPLATE;
  DELETE definition-path< / STORE=libref.template-store>;
  LINK definition-path-1 TO definition-path-2 <option(s)>;
  LIST <starting-path><option(s)>;
  PATH location(s);
  SOURCE definition-path <option(s)><STORE=libref.template-store>;
  TEST DATA=SAS-data-set< / STORE=libref.template-store>;
```

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete the specified definition</td>
<td>“DELETE Statement” on page 273</td>
</tr>
<tr>
<td>Create a link to an existing definition</td>
<td>“LINK Statement” on page 273</td>
</tr>
<tr>
<td>List items in one or more template stores</td>
<td>“LIST Statement” on page 274</td>
</tr>
<tr>
<td>Specify which locations to write to or read from when you create or use PROC TEMPLATE definitions, and specify the order in which to search for them</td>
<td>“PATH Statement” on page 276</td>
</tr>
<tr>
<td>Write the source code for the specified definition to the SAS log</td>
<td>“SOURCE Statement” on page 277</td>
</tr>
<tr>
<td>Test the most recently created definition by binding it to the specified data set</td>
<td>“TEST Statement” on page 279</td>
</tr>
</tbody>
</table>

---

**PROC TEMPLATE Statement**

```sas
PROC TEMPLATE;
```
DELETE Statement

Deletes the specified definition

```
DELETE definition-path;
```

Required Arguments

definition-path

specifies a definition to delete. A definition-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 definition exists in multiple template stores, PROC TEMPLATE deletes the definition 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 definitions in the directory. If the path that you specify is a directory rather than a definition, PROC TEMPLATE deletes all the directories and all the definitions in that directory.

LINK Statement

Creates a link to an existing definition

```
LINK definition-path-1 TO definition-path-2 / option(s);
```

Creating a link to a definition has the same effect as creating a new definition that inherits its characteristics from another definition (see the discussion of PARENT= on page 419 option). However, using a link is more efficient than using inheritance because linking does not actually create a new definition.

**Note:** To maximize efficiency, PROC TEMPLATE implements any definition that consists solely of the declaration of a parent and of notes as a link.

Required Arguments

definition-path-1

specifies the path of the definition to create. PROC TEMPLATE creates the definition in the first template store in the path that you can write to.

definition-path-2

specifies the path of the definition to link to. If the same definition 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 definition-path-2 exists when it compiles the definition.

Options

NOTES= 'text'
specifies notes to store in the definition.

Requirement: You must enclose the text in quotation marks.

Tip: Notes of this type become part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

STORE=libref.template-store
specifies the location where the link will be created.

Restriction: The STORE= option syntax does not become part of the compiled definition.

Tip: The link always points to the first definition with the same name that it finds in the ODS path.

LIST Statement

Lists the definitions in one or more template stores

Featured in: Example 1 on page 281

LIST <starting-path><option(s)>;

Options

starting-path
specifies a level within each template store where PROC TEMPLATE starts listing definitions. 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 do not specify a starting-path, then the LIST statement lists all definitions in all template stores unless the ODS PATH statement is used to confine the search to the specified template stores.

Restriction: This option must precede the forward slash (/) in the LIST statement.

SORT=statistic <sorting-order>
sorts the list of definitions by the specified statistic in the specified sorting order.

statistic
  can be one of the following:

  CREATED
  is the date that the definition 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 definition that the current definition links to (see “LINK
  Statement” on page 273).

PATH
  is the path to the current definition in the template store. (The path does not
  include the name of the template store).

SIZE
  is the size of the definition.

TYPE
  is the type of definition: COLUMN, FOOTER, HEADER, STYLE, TABLE, or
  LINK. If the item is not a definition, but simply a level in the item store, its
  type is DIR.

  Default: PATH

  sorting-order
  specifies whether SORT= sorts from low values to high values or from high values
  to low values.

  ASCENDING
    sorts from low values to high values.

    Alias: A

  DESCENDING
    sorts from high values to low values.

    Alias: D

  Default: ASCENDING

STATS=ALL | (statistic-1 <, ... statistic-n>)
  specifies the information to include in the list of definitions.

  ALL
    includes all available information.

(statistic-1 <, ... statistic-n>)
  includes the specified information. statistic can be one or more of the following:

  CREATED
    is the date that the definition 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 definition that the current definition links to (see “LINK
  Statement” on page 273).

SIZE
  is the size of the definition.
Default: Whether or not you specify STATS=, the list of definitions always includes an observation number, the path to the definition, and its type.

STORE=libref.template-store
specifies the template store to process.
Default: all template stores in the current template path (see “PATH Statement” on page 276).

---

PATH Statement

Specifies locations to write to or read from when you create or use PROC TEMPLATE definitions, and specifies the order in which to search for them. This statement overrides the ODS PATH statement for the duration of the PROC TEMPLATE step.

Featured in: Example 1 on page 281 and Example 2 on page 283

PATH <(APPEND) | (PREPEND) | (REMOVE) > location(s);
PATH path-argument;

Required Arguments

location(s)
specifies one or more locations to write to or read from when creating or using PROC TEMPLATE definitions and the order in which to search for them. ODS searches the locations in the order that they appear on the statement. It uses the first definition that it finds that has the appropriate access mode (read, write, or update) set.

Each location has the following form:
<libref.item-store <(READ | UPDATE | WRITE)>  

<libref.item-store
identifies an item store to read from, to write to, or to update. If an item store does not already exist, then the PATH statement will create it.

(READ | UPDATE | WRITE)
specifies the access mode for the definition. 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:**

SASUSER.TEMPLAT (UPDATE)
SASHELP.TMPLMST (READ)

*Note:* SAS stores all the definitions that it provides in SASHELP.TMPLMST.

**Tip:** If you want to be able to ignore all the definitions that you create, then keep them in their own item stores so that you can leave them out of the list of item stores that ODS searches.

**path-argument**

sets or displays the ODS path.

*path-argument* can be one of the following:

- **RESET**
  sets the ODS path to the default settings SASUSER.TEMPLAT (UPDATE) and SASHELP.TMPLMST (READ).

- **SHOW**
  displays the current ODS path.

- **VERIFY**
  sets the ODS path to include only templates supplied by SAS. VERIFY is the same as specifying ODS PATH SASHELP.TMPLMST (READ).

**Options**

**(APPEND | PREPEND | REMOVE)**

adds 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 are 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 are kept.

- **REMOVE**
  removes one or more locations from a path.

*Default:* If you do not specify an APPEND, PREPEND, or REMOVE option, then the PATH statement overwrites the complete path.

---

**SOURCE Statement**

Writes the source code for the specified definition to the SAS log

**Featured in:** Example 2 on page 283
SOURCE definition-path <option(s)>;

Required Arguments

definition-path
specifies the path of the definition that you want to write to the SAS log. If the same definition 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 definitions in compiled form. The SOURCE statement actually decompiles the definition. Because SAS comments are not compiled, comments that are in the source code do not appear when you decompile the definition. If you want to annotate your definition, use the NOTES statement inside the definition or the block of editing instructions, or use the NOTES= option in the LINK statement. These notes do become part of the compiled definition. (See “NOTES Statement” on page 429 and the discussion of the NOTES= option on page 274. You can also specify notes as quoted strings in the DYNAMIC, MVAR, NLMVAR, REPLACE, and STYLE statements.)

Options

FILE= 'file-specification' | fileref
specifies a file to write the definition 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. (For information on the FILENAME statement, see “Statements” in SAS Language Reference: Dictionary.)

**Default:** If you do not specify a filename where you want the source code written, then the SOURCE statement writes the source code to the SAS log.

NOFOLLOW
specifies that the program not resolve links in the PARENT= option, which specifies the definition that the current definition inherits from. For information about the PARENT= option, see the PARENT= option in the styles attribute section.

STORE= libref.template-store
specifies the template store where the definition is located.

**Interaction:** In most cases, the STORE= option is added to the definition statement when PROC TEMPLATE displays the source code. However, if the template store specified in the STORE= option is in the ODS path with only read permission, then PROC TEMPLATE does not include the STORE= option in the source code that it displays. There will be no STORE= option, which means that if you run the code, then the definition that it creates will go to the first template store in your ODS path that has update permission.
TEST Statement

Tests the most recently created definition by binding it to the specified data set

\[
\text{TEST DATA=} \text{SAS-data-set} < / \text{STORE=} \text{libref.template-store}>;
\]

Required Arguments

\text{DATA=} \text{SAS-data-set} \\
specifies the data set to bind to the most recently created definition. ODS sends this output object to all open ODS destinations.

Options

\text{STORE=} \text{libref.template-store} \\
specifies the template store where the definition is located. \\
\textbf{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 definition.

Concepts: Template Stores and the TEMPLATE Procedure

The Contents of Definitions (Templates) that SAS Supplies

SAS provides definitions (templates) for these items:
- tables
- styles
- tagsets

To view the contents of a definition (template), you can use the SAS windowing environment, the SAS window command \textit{odstemplates}, or the TEMPLATE procedure.

- **SAS Windowing Environment**
  1. From the SAS Explorer, select \textit{View} \rightarrow \textit{Results}
  2. In the Results window, select the Results folder. Right click to open the Templates window.
  3. To view the definitions (templates) that SAS supplies, click on the plus sign that is next to the SASHELP.TMPLMST item store.
  4. Click on 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.
Display 8.1 Definitions (Templates) that SAS Supplies

SAS Windowing Command

1. To view the Templates window, submit the following command in the command bar:
   
   `ods_templates`

   The following display shows the Templates window that contains the item stores `Sasuser.Templat` and `Sashelp.Tmplmst`.

2. When you double-click an item store, such as `Sashelp.Tmplmst`, that item store expands to list the directories where ODS templates are stored. The templates that SAS provides are in the item store `Sashelp.Tmplmst`. 
The SOURCE statement writes the source code for the specified definition to the SAS log. For example, if you want to view the source for all the objects in Base SAS, submit the following code.

```sas
proc template;
source base;
run;
```

*Note:* For more information, see “SOURCE Statement” on page 277.

---

### Examples: Managing Template Stores Using TEMPLATE Procedure

#### Example 1: Listing Definitions in a Template Store

PROC TEMPLATE features:
- PATH statement
- LIST statement
  - `starting-path` option
  - `SORT=` option

---

**Program Description**

This example lists the items for the Base.Univariate directory in the item store SASHELP.TMPLMST.
**Set the SAS system options.** The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output.

```sas
options nodate pageno=1 pagesize=60 linesize=72;
```

**Specify which locations to search for definitions that were created by PROC TEMPLATE.** The PATH statement specifies to search for definitions that were created by PROC TEMPLATE in the SASHELP.TMPLMST item store.

```sas
proc template;
  path sashelp.tmplmst;
run;
```

**List in descending order the definitions that are stored within a specified level of the template store.** The LIST statement lists the 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 definitions. The SORT= option sorts the list of definitions. The definitions are sorted in descending order.

```sas
list base.univariate / sort=path descending;
run;
```

**Display 8.3** Listing of `Base.Univariate` Template Store

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Base.Univariate.Wins</td>
<td>Table</td>
</tr>
<tr>
<td>2</td>
<td>Base.Univariate.Trim</td>
<td>Table</td>
</tr>
<tr>
<td>3</td>
<td>Base.Univariate.RobustScal</td>
<td>Table</td>
</tr>
<tr>
<td>4</td>
<td>Base.Univariate.Quantiles</td>
<td>Table</td>
</tr>
<tr>
<td>5</td>
<td>Base.Univariate.PValue</td>
<td>Link</td>
</tr>
<tr>
<td>6</td>
<td>Base.Univariate.Normal</td>
<td>Table</td>
</tr>
<tr>
<td>7</td>
<td>Base.Univariate.Moments</td>
<td>Table</td>
</tr>
<tr>
<td>8</td>
<td>Base.Univariate.Modes</td>
<td>Table</td>
</tr>
<tr>
<td>9</td>
<td>Base.Univariate.Missing</td>
<td>Table</td>
</tr>
<tr>
<td>10</td>
<td>Base.Univariate.Measures</td>
<td>Table</td>
</tr>
<tr>
<td>11</td>
<td>Base.Univariate.Location</td>
<td>Table</td>
</tr>
<tr>
<td>12</td>
<td>Base.Univariate.LocCount</td>
<td>Table</td>
</tr>
<tr>
<td>13</td>
<td>Base.Univariate.Frequency</td>
<td>Table</td>
</tr>
<tr>
<td>14</td>
<td>Base.Univariate.FitQuant</td>
<td>Table</td>
</tr>
<tr>
<td>15</td>
<td>Base.Univariate.FitParsm</td>
<td>Table</td>
</tr>
<tr>
<td>16</td>
<td>Base.Univariate.FitGood</td>
<td>Table</td>
</tr>
<tr>
<td>17</td>
<td>Base.Univariate.ExtVal</td>
<td>Table</td>
</tr>
<tr>
<td>18</td>
<td>Base.Univariate.ExtObs</td>
<td>Table</td>
</tr>
<tr>
<td>19</td>
<td>Base.Univariate.ConfLimits</td>
<td>Table</td>
</tr>
<tr>
<td>20</td>
<td>Base.Univariate.Bins</td>
<td>Table</td>
</tr>
<tr>
<td>21</td>
<td>Base.Univariate.BinPercents</td>
<td>Table</td>
</tr>
<tr>
<td>22</td>
<td>Base.Univariate</td>
<td>Dir</td>
</tr>
</tbody>
</table>
Example 2: Viewing the Source of a Definition

PROC TEMPLATE features:
   PATH statement
   SOURCE statement

Program Description

This example displays the source code for the tagset definition Xhtml that SAS provides.

Program

Specify which locations to search for definitions that were created by PROC TEMPLATE. The PATH statement specifies to search for definitions that were created by PROC TEMPLATE in the SASHELP.TMPLMST item store.

```
proc template;
   path sashelp.tmplmst;
```

Write the source code of the specified definition. The SOURCE statement writes the source code for the tagset Xhtml that SAS provides. The source code is written to the SAS log.

```
source Tagsets.Xhtml;
run;
```

Display 8.4 Source Code of the Definition Tagset Xhtml That Is Written to the SAS Log

```
NOTE: Path 'Tagsets.Xhtml' is in: SASHELP.TMPLMST.
define tagset Tagsets.Xhtml;
   notes "XHTML 1.0";
   define event doc;
     start:
       set empty_tag_suffix "";
       set $doctype
           <!DOCTYPE html PUBLIC ""\n3C//DTD XHTML 1.0 Transition\n1//EN"">;
       set $framesetdoctype
           <!DOCTYPE html PUBLIC ""\n3C//DTD XHTML 1.0 Frameset\nEN"">;
     put $doctype NL;
     put "<html>" NL;
     finish:
     put "</html>" NL;
   end;
   split = "<br/>";
   parent = tagsets.html4;
end;
run;
```

NOTE: PROCEDURE TEMPLATE used (Total process time):
   real time    0.10 seconds
   cpu time     0.11 seconds
Overview: ODS Style Definitions

Why Use the TEMPLATE Procedure to Create a Style Definition?

The TEMPLATE procedure enables you to customize the look of your SAS output. The TEMPLATE procedure creates and modifies style definitions. The Output Delivery System then uses these style definitions to produce customized formatted output.
By default, ODS output is formatted according to the various style definitions that the procedure or DATA step specify. However, you can also customize the appearance of your output by using the DEFINE STYLE statement in the TEMPLATE procedure.

## Terminology

For definitions of terms used in this section, see “Terminology: TEMPLATE Procedure” on page 266.

### What Can You Do with a Style Definition?

#### Default Style Definition for HTML

By default, ODS uses style definitions to display the procedure or DATA step results. You can modify the appearance of your output by customizing these style definitions. Display 9.1 on page 286 shows the HTML output from PROC PRINT using the default style definition. Display 9.2 on page 287 shows the same HTML output from PROC PRINT with a customized style definition.

#### Display 9.1 HTML Output from PROC PRINT That Uses the Default Style Definition (Viewed with Microsoft Internet Explorer)

![HTML Output](image)

**Energy Expenditures for Each Region**

* (millions of dollars)

<table>
<thead>
<tr>
<th>Division=Middle Atlantic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
</tr>
<tr>
<td>NY</td>
</tr>
<tr>
<td>NY</td>
</tr>
<tr>
<td>NJ</td>
</tr>
<tr>
<td>NJ</td>
</tr>
<tr>
<td>PA</td>
</tr>
<tr>
<td>PA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Division=Mountain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
</tr>
<tr>
<td>MT</td>
</tr>
<tr>
<td>MT</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>ID</td>
</tr>
</tbody>
</table>

#### Customized Version of the HTML Style Definition

When you are working with style definitions, you are more likely to modify a SAS style definition than to write a completely new style definition. The next display shows the kinds of changes that you can make to the default style definition for the HTML output. The new style definition affects both the contents file and the body file in the HTML output. In particular, in the contents file, the style definition makes changes to
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 headers. 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 such as cell spacing, rules, and border width.

In the body file, the new style definition makes changes to
- the text of the header and the text that identifies the procedure that produced the output
- the colors for some parts of the text
- the font size of some parts of the text
- the spacing in the list of entries in the table of contents.

Display 9.2  HTML Output from PROC PRINT with the Customized Style Definition  (Viewed with Microsoft Internet Explorer)

Style Syntax:  TEMPLATE Procedure

PROC TEMPLATE;
PROC TEMPLATE Statement

PROC TEMPLATE;
DEFINE STYLE style-path <! STORE=libref.template-store>; 
statements-and-attributes 
END;

DEFINE STYLE Statement

Creates a style definition for any destination that supports the STYLE= option

Requirement: An END statement must be the last statement in the definition.
Featured in: Example 1 on page 342

DEFINE STYLE style-path <! STORE=libref.template-store>; 
<PARENT=style-path;>
NOTES 'text';
REPLACE new-style-element-name <FROM existing-style-element-name><'text'> 
< / style-attribute-specification(s)>;
STYLE new-style-element-name <FROM existing-style-element-name><'text'> 
< / style-attribute-specification(s)>;
END;

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide information about the style definition.</td>
<td>NOTES</td>
</tr>
<tr>
<td>Add a style element to the child style definition from the parent style definition.</td>
<td>REPLACE</td>
</tr>
<tr>
<td>Create a new style element.</td>
<td>STYLE</td>
</tr>
<tr>
<td>End the style definition.</td>
<td>END</td>
</tr>
</tbody>
</table>

Required Arguments

*style-path*
specifies where to store the style definition. 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 definition to the first template store that you can write to in the current path.

Options

STORE=libref.template-store
specifies the template store in which to store the definition. 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 definition.

Style Definition Attributes

PARENT=style-path
specifies the style definition for which the current definition is inherited from. A style-path consists of one or more names, separated by periods. Each name represents a directory in a template store. The current style definition inherits from the specified style definition in the first template store that you can read from in the current path.

When you specify a parent, all the style elements, style attributes, and statements that are specified in the parent’s definition are used in the current definition unless the current definition overrides them.

SAS provides some style definitions. You can specify one of these style definitions for style-path, or you can specify a user-defined style definition. Some of the style definitions that are currently shipped with SAS include:

- styles.default
- styles.beige
- styles.brick
- styles.brown
- styles.d3d
- styles.minimal
- styles.printer
- styles.statdoc.

For information about finding an up-to-date list of the style definitions and for viewing a style definition, see “Viewing the Contents of a Style Definition” on page 319.

NOTES Statement

Provides information about the style definition

Tip: The NOTES statement becomes part of the compiled style definition, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES 'text';
Required Arguments

\textit{text}

provides information about the style definition.

---

**REPLACE Statement**

Adds a style element to the child style definition from the parent style definition

**Restriction:** To use the REPLACE statement, you must specify a parent style definition with the \texttt{PARENT=} attribute in the DEFINE STYLE statement.

**See also:** “About Style Definition Inheritance and Style Element Inheritance” on page 322

**Featured in:** Example 3 on page 355

**Tip:** You can think of the REPLACE statement as replacing the statement that defines the like-named style element in the parent style definition. The REPLACE statement does not actually change the parent style definition, but PROC TEMPLATE builds the child style definition as if it had changed the parent. All style elements that inherit attributes from this style element inherit the attributes that are specified in the REPLACE statement, not those used in the parent style definition.

\begin{verbatim}
REPLACE style-element-name-1 <FROM style-element-name-2><\textit{text}>
  </style-attribute-specification(s)>;
\end{verbatim}

**Required Arguments**

\textit{style-element-name-1}

names the style element to replace. A like-named style element must exist in the parent style definition. PROC TEMPLATE stores \textit{style-element-name-1} in the current style definition and replaces all its attributes with the attributes that you specify in the REPLACE statement. If an attribute is defined in the like-named style element in the parent style definition and you do not explicitly specify it in the REPLACE statement, then the value of the attribute defaults to the value that was inherited from the parent of the like-named style element.

**Options**

\textit{style-element-name-2}

names the style element that \textit{style-element-name-1} inherits from. The style element must exist in the current style definition or in the parent of the current style definition. PROC TEMPLATE looks first in the current style definition for the style element. If PROC TEMPLATE does not find the style element, then it looks in the parent style definition.
style-attribute-specification(s) specifies the style attributes for style-element-name-1. The new style element inherits from the parent style element all the attributes that the parent inherits. However, all the attributes that are explicitly specified in the definition of style-element-name-2 must be respecified in the REPLACE statement if you want to keep them. You can override any attribute of the parent style element, whether it is inherited or explicitly defined, by specifying it in the REPLACE statement. Each style-attribute-specification has the following general form:

style-attribute-name=style-attribute-value

style-attribute-name can be the name of an attribute that is listed in “Style Attributes and Their Values” on page 292, or it can be the name of a user-defined attribute.

Restriction: If style-attribute-name refers to a user-defined attribute, then you must enclose the name in quotation marks. If style-attribute-name refers to an attribute that is listed in “Style Attributes and Their Values” on page 292, then you do not enclose the name in quotation marks. For more information about user-defined attributes, see “Style Attributes and Their Values” on page 292.

style-attribute-value assigns the value to the attribute. For information about style-attribute values, see “Style Attributes and Their Values” on page 292.

'text' provides information about the REPLACE statement. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

---

**STYLE Statement**

Creates a new style element

Featured in: Example 1 on page 342

STYLE new-style-element-name <FROM existing-style-element-name> ’text’
</style-attribute-specification(s)>;

**Required Arguments**

new-style-element-name names the style element to create. PROC TEMPLATE stores the style element in the current style definition.

**Options**

existing-style-element-name names an existing style element to inherit from. The style element must exist in the current style definition or in the parent of the current style definition.
**style-attribute-specification(s)**

specify new style attributes or modifications to existing style attributes for the new style element. The new style element inherits all of the style attributes of `existing-style-element-name`. You can override any of these attributes by specifying attributes in the STYLE statement. Each `style-attribute-specification` has the following general form:

```
style-attribute-name=style-attribute-value
```

- **style-attribute-name**
  can be the name of an attribute that is listed in “Style Attributes and Their Values” on page 292, or it can be the name of a user-defined style attribute.

- **Restriction**: If `style-attribute-name` refers to a user-defined attribute, then you must enclose the name in quotation marks. If `style-attribute-name` refers to an attribute that is listed in “Style Attributes and Their Values” on page 292, then do not enclose the name in quotation marks.

- **style-attribute-value**
  assigns the value to the attribute. If you use an attribute from the list in “Style Attributes and Their Values” on page 292, then you must use the kind of value that the attribute expects.

  For more information about style-attribute values, see “Style Attributes and Their Values” on page 292.

---

`text`

provides information about the REPLACE statement. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

---

**Style Attributes and Their Values**

The default value that is used for an attribute depends on the style definition that is in use. For information about viewing the attributes in a style definition, see “Viewing the Contents of a Style Definition” on page 319. The implementation of an attribute depends on the ODS destination that formats the output. In addition, if you are creating HTML output, then the implementation of an attribute depends on the browser that you use.

**Data Values**

Values for style attributes are one of the following:

- **color**
  is a string that identifies a color. A color can be
  - any of the color names that are supported by SAS/GRAPH. These names include
    - a predefined SAS color (for example, blue or VIYG)
    - a red/green/blue (RGB) value (for example, CX0023FF)
    - a hue/light/saturation (HLS) value (for example, H14E162D)
    - a gray-scale value (for example, GRAYBB).
  - an RGB value with a leading pound sign (#) rather than CX (for example, #0023FF).
  - one of the colors that exists in the SAS session when the style definition is used:
    - DMSBLUE
Note: Use these colors only if you are 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). You can use the Color Naming System to form a color by 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.

The words that you can use are shown in the following table:

<table>
<thead>
<tr>
<th>Lightness</th>
<th>Saturation</th>
<th>Chromatic Hue</th>
<th>Achromatic Hue</th>
</tr>
</thead>
<tbody>
<tr>
<td>very dark</td>
<td>grayish</td>
<td>purple</td>
<td>black *</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 gray **</td>
</tr>
<tr>
<td>light</td>
<td>vivid</td>
<td>yellow</td>
<td></td>
</tr>
<tr>
<td>very light</td>
<td>vivid</td>
<td>green</td>
<td></td>
</tr>
</tbody>
</table>

| white * |

* Black and white cannot be combined with a lightness or a saturation value.
** Gray cannot be combined with a saturation value.

You can combine these words to form a wide variety of colors. Some examples are

- 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 the following adjectives with one of its neighboring colors:

- reddish
- orangish
- brownish
- yellowish
- greenish
- bluish
- purplish.

For example, you can use the following as hues:

- bluish purple
- reddish orange
- yellowish green.

See also: SAS/GRAPH Reference, Volumes 1 and 2 for information about SAS/GRAPH colors.

dimension

is a nonnegative number, optionally followed by one of the following units of measure:

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

Default: For the Printer destination, units of 1/150 of an inch

font-definition

is the name of a font, the font size, and font keywords. A font definition has the following general format:

("font-face-1 <… , font-face-n>", font-size, keyword-list)

If you specify only one font and if its name does not include a space character, then you can omit the quotation marks. If you specify more than one font, then the destination device uses the first one that is installed on your system.

font-size specifies the size of the font. font-size can be a dimension or a number without units of measure. If you specify a dimension, then you must 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 document. For more information see dimensions on page 294.
**keyword-list** specifies the font weight, font style, and font width. You can include one value for each, in any order. The following table shows the keywords that you can use:

<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>NORMAL</td>
</tr>
<tr>
<td>BOLD</td>
<td>ROMAN</td>
<td>COMPRESSED</td>
</tr>
<tr>
<td>DEMI_BOLD</td>
<td>SLANT</td>
<td>EXTRA_COMPRESSED</td>
</tr>
<tr>
<td>EXTRA_BOLD</td>
<td></td>
<td>NARROW</td>
</tr>
<tr>
<td>LIGHT</td>
<td></td>
<td>WIDE</td>
</tr>
<tr>
<td>DEMI_LIGHT</td>
<td></td>
<td>EXPANDED</td>
</tr>
<tr>
<td>EXTRA_LIGHT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Few fonts honor these values.

**Featured in:** Example 2 on page 348

**format**

is a SAS format or a user-defined format.

**reference**

is a reference to an attribute that is defined in the current style definition or in the parent style definition (or beyond). The value that you use is the name of the style element followed by the name of an attribute, in parentheses, within that element. For example, suppose that you create a style element called DATACELL that uses the FOREGROUND= and BACKGROUND= style elements this way:

```sas
style datacell / background=blue
foreground=white;
```

Later, you can ensure that another style element, NEWCELL, uses the same background color by defining it this way:

```sas
style newcell / background=datacell(background);
```

Similarly, suppose that you create a style element called HIGHLIGHTING that defines three attributes this way:

```sas
style highlighting /
"go"=green
"caution"=yellow
"stop"=red;
```

Later, you can define a style element called MESSAGES that uses the colors that are defined in HIGHLIGHTING:

```sas
style messages;
"note"=highlighting("go")
"warning"=highlighting("caution")
"error"=highlighting("stop");
```

In this way, multiple style elements could use the colors that you define in HIGHLIGHTING. If you decide to change the value of go to blue, you simply change its value in the definition of HIGHLIGHTING, and every style element that references highlighting (“go”) will use blue instead of green.
Note: In the first example, the style attribute BACKGROUND= is a predefined style attribute. Therefore, when you reference it, you do not put it in quotation marks. However, in the second example, go is a user-defined attribute. You define it with quotation marks, and when you reference it, you must use quotation marks.

You can use a special form of reference to get a value for a style attribute from the macro table at the time that the style element is used. For example, the following STYLE statement uses the current value of the macro variable bkgr for the background color of the style element cell:

```plaintext
style cell / background=symget(“bkgr”);
```

**Featured in:** Example 2 on page 348

'\textit{string}'
is a quoted character string.

**Style Attributes**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Influence the characteristics of individual cells</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify how to handle leading spaces and line breaks.</td>
<td>ASIS=</td>
<td>HTML, RTF, PDF, PCL, and PS</td>
</tr>
<tr>
<td>Specify the height of the cell.</td>
<td>CELLEIGHT=</td>
<td>HTML, RTF, PDF, PCL, and PS</td>
</tr>
<tr>
<td>Specify the width of the cell.</td>
<td>CELLWIDTH=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the text to show in a tool tip for the cell.</td>
<td>FLYOVER=</td>
<td>HTML, PDF</td>
</tr>
<tr>
<td>Specify the window or frame in which to open the target of the link.</td>
<td>HREFTARGET=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify how to handle space characters.</td>
<td>NOBREAKSPACE=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify text to insert in the HTML</td>
<td>TAGATTR=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify a URL to link to.</td>
<td>URL=</td>
<td>HTML, RTF, and PDF</td>
</tr>
<tr>
<td>Specify vertical justification.</td>
<td>VJUST=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
</tbody>
</table>

**Influence the characteristics of individual tables or cells**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the color of the background.</td>
<td>BACKGROUND=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify an image to use as the background.</td>
<td>BACKGROUNDIMAGE=</td>
<td>HTML, PCL and PS</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Valid destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Specify the color of the border if the border is just one color.</td>
<td>BORDERCOLOR</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the darker color to use in a border that uses two colors to create a three-dimensional effect.</td>
<td>BORDERCOLORDARK</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the lighter color to use in a border that uses two colors to create a three-dimensional effect.</td>
<td>BORDERCOLORLIGHT</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the width of the border of the table.</td>
<td>BORDERWIDTH</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify a font definition.</td>
<td>FONT=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the font to use.</td>
<td>FONT_FACE=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the size of the font.</td>
<td>FONT_SIZE=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the style of the font.</td>
<td>FONT_STYLE=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the font weight.</td>
<td>FONT_WEIGHT=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the font width compared to the width of the usual design.</td>
<td>FONT_WIDTH=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the color of the foreground, which is primarily the color of the text.</td>
<td>FOREGROUND=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the name of the stylesheet class to use for the table or cell.</td>
<td>HTMLCLASS=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify an ID for the table or cell.</td>
<td>HTMLID=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify individual attributes and values for the table or cell.</td>
<td>HTMLSTYLE=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify justification.</td>
<td>JUST=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the HTML code to place after the HTML table or cell.</td>
<td>POSTHTML=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify an image to place after the table or cell.</td>
<td>POSTIMAGE=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Valid destinations ...</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Specify text to place after the cell or table.</td>
<td>POSTTEXT=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the HTML code to place before the HTML table or cell.</td>
<td>PREHTML=</td>
<td>HTML</td>
</tr>
<tr>
<td>Specify an image to place before the table or cell.</td>
<td>PREIMAGE=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify text to place before the cell or table.</td>
<td>PRETEXT=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Determine how less-than signs (&lt;), greater-than signs (&gt;) and ampersands (&amp;) are interpreted.</td>
<td>PROTECTSPECIALCHARACTERS=</td>
<td>HTML, MARKUP family, Printer family, and RTF</td>
</tr>
</tbody>
</table>

**Influence the characteristics of tables**

| Specify the amount of white space on each of the four sides of the text in a cell. | CELLPADDING= | HTML, PCL, PDF, PS, and RTF |
| Specify the thickness of the spacing between cells.                    | CELLS spacing= | HTML, PCL, PDF, PS, and RTF |
| Specify the type of frame to use on an HTML table.                     | FRAME=         | HTML, PRINTER family, and RTF |
| Specify the width of the table.                                         | OUTPUTWIDTH=   | HTML, PCL, PDF, PS, and RTF     |
| Specify the types of rules to use in a table.                          | RULES=         | HTML, PCL, PDF, PS, and RTF     |

**Influence the characteristics of individual frames in HTML output**

| Specify whether or not to put a scrollbar in the frame that references the body file. | BODYSCROLLBAR= | HTML |
| Specify the width of the frame that displays the body file in the HTML frame file. | BODYSIZE=      | HTML |
| Specify the string to use for bullets in the contents file.             | BULLETS=       | HTML |
| Specify the position of the frames in the frame file that displays the contents and the page files. | CONTENTPOSITION= | HTML |
| Specify whether or not to put a scrollbar in the frames in the frame file that displays the contents and the page files. | CONTENTSCROLLBAR= | HTML |
### Task

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTENTSIZE=</td>
<td>HTML</td>
</tr>
<tr>
<td>FRAMEBORDER=</td>
<td>HTML</td>
</tr>
<tr>
<td>FRAMEBORDERWIDTH=</td>
<td>HTML</td>
</tr>
<tr>
<td>FRAMESPACING=</td>
<td>HTML</td>
</tr>
</tbody>
</table>

#### Influence the characteristics of the document

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT=</td>
<td>HTML and MARKUP</td>
</tr>
<tr>
<td>ACTIVELINKCOLOR=</td>
<td>HTML and RTF</td>
</tr>
<tr>
<td>BOTTOMMARGIN=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>FILLRULEWIDTH=</td>
<td>PS, PDF, PCL</td>
</tr>
<tr>
<td>HTMLCONTENTTYPE=</td>
<td>HTML</td>
</tr>
<tr>
<td>HTMLDOCTYPE=</td>
<td>HTML</td>
</tr>
<tr>
<td>INDENT=</td>
<td>MARKUP, RTF and PRINTER Family</td>
</tr>
<tr>
<td>LEFTMARGIN=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>LINKCOLOR=</td>
<td>HTML, RTF, and PDF</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Specify whether or not to make this entry in the table of contents a link to the body file.</td>
<td>LISTENTRYANCHOR=</td>
</tr>
<tr>
<td>Specify whether or not to double space between entries in the table of contents.</td>
<td>LISTENTRYDBLSPACE=</td>
</tr>
<tr>
<td>Specify the height for graphics in the document.</td>
<td>OUTPUTHEIGHT=</td>
</tr>
<tr>
<td>Specify an upper limit for extending the width of the column.</td>
<td>OVERHANGFACTOR=</td>
</tr>
<tr>
<td>Specify HTML to place at page breaks.</td>
<td>PAGEBREAKHTML=</td>
</tr>
<tr>
<td>Specify the right margin for the document.</td>
<td>RIGHTMARGIN=</td>
</tr>
<tr>
<td>Specify the top margin for the document.</td>
<td>TOPMARGIN=</td>
</tr>
<tr>
<td>Specify the color for links the visited links.</td>
<td>VISITEDLINKCOLOR=</td>
</tr>
<tr>
<td>Specify whether or not to make the image that is specified by BACKGROUNDIMAGE= into a “watermark.” A watermark appears in a fixed position as the window is scrolled.</td>
<td>WATERMARK=</td>
</tr>
</tbody>
</table>

*Influence the characteristics of graphs*

<p>| Specify the background color of the graph. | BACKGROUND=                                 | HTML, RTF, PRINTER family |
| Specify the image to appear in the background. This image will be stretched. | BACKGROUNDIMAGE=                            | HTML, PCL, and PS |
| Specify the alternate colors for maps. The alternate colors are applied to the blocks on region areas in block maps. | CONTRASTCOLOR=                              | HTML, RTF, PRINTER family |
| Specify whether to use a drop shadow effect for text in a graph.     | DROPSHADOW=                                 | HTML, RTF, PRINTER family |
| Specify the end color for a gradient effect in a graph.              | ENDCOLOR=                                   | HTML, RTF, PRINTER family |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a font definition. 1</td>
<td>FONT=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the font to use. 1</td>
<td>FONT_FACE=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the size of the font to use. 1</td>
<td>FONT_SIZE=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the style of the font. 1</td>
<td>FONT_STYLE=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the font weight. 1</td>
<td>FONT_WEIGHT=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the font width compared to the width of the usual design. 1</td>
<td>FONT_WIDTH=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the color of text or data items 1</td>
<td>FOREGROUND=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the direction of the gradient effect in either the X or Y axis direction to influence the graph background, legend background, charts, walls, floors, etc.</td>
<td>GRADIENT_DIRECTION=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the image to appear in the background. This image can be positioned or tiled.</td>
<td>IMAGE=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the image's horizontal positioning. 1</td>
<td>JUST=</td>
<td>HTML, PCL, PDF, PS, and RTF</td>
</tr>
<tr>
<td>Specify the line type to use in a graph. You can use SAS/GRAPH line types 1–46.</td>
<td>LINESTYLE=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the thickness (width) of a line that is part of a graph.</td>
<td>LINETHICKNESS=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the size of the symbol used to represent data values.</td>
<td>MARKERSIZE=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the symbol used to represent data values.</td>
<td>MARKERSYMBOL=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the height of the graph. 1</td>
<td>OUTPUTHEIGHT=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the width of the graph or line thickness. 1</td>
<td>OUTPUTWIDTH=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the start color for a gradient effect in a graph.</td>
<td>STARTCOLOR=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Valid destinations</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Specify the level of transparency for a graph.</td>
<td>TRANSPARENCY=</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
<tr>
<td>Specify the image's vertical positioning.</td>
<td>VJUST</td>
<td>HTML, RTF, PRINTER family</td>
</tr>
</tbody>
</table>

1 This attribute can also be used to influence other characteristics as described in another section of the table.

**Note:** You can use the value _UNDEF_ for any style attribute. ODS treats an attribute that is set to _UNDEF_ as if its value had never been set, even in the parent or beyond. △

**ABSTRACT= ON | OFF**
determines whether or not styles are used in CSS or LaTex style files.

- **ON** specifies that styles are used in CSS or LaTex style files.
- **OFF** specifies that styles are not used in CSS or LaTex style files.

**Applies to:** document
**ODS Destination:** HTML, MARKUP, and LaTex

**ACTIVELINKCOLOR=**
specifies the color that a link changes to after you click on it, but before the browser opens that file.

**Applies to:** document
**ODS Destination:** HTML

**See:** color on page 292

**ASIS=ON | OFF**
specifies how to handle leading spaces and line breaks.

- **ON** prints text with leading spaces and line breaks, in the same manner as the listing output.
- **OFF** trims leading spaces and ignores line breaks.

**Default:** OFF
**Applies to:** document
**ODS Destinations:** HTML, RTF, PS, PCL, and PDF

**BACKGROUND=**
specifies the color of the background.

**Tip:** Generally, the background color of the cell overrides the background color of the table. You see the background color for the table only as the space between cells (see CELLSPACING= on page 305).

**Applies to:** tables or cells and graphs
**ODS Destinations:** HTML, PCL, PDF, PS, and RTF

**Overridden by:** CBACK= option in the SAS/GRAPH GOPTIONS statement

**Featured in:** Example 1 on page 342 and Example 3 on page 355
See: color on page 292

BACKGROUNDIMAGE='string'
specifies an image to use as the background. Viewers can tile or stretch the image as the background for the HTML table or graph that the procedure creates. For graphs, the specified image is stretched. *string* is the name of a GIF or JPEG file. You can use a simple file name, a complete path, or a URL. However, the most versatile approach is to use a simple filename and to place all image files in the local directory.

 Applies to: tables or cells and graphs

ODS Destinations: HTML, PCL, and PS

 Overridden by: IBACK= and IMAGESTYLE=FIT options in the SAS/GRAPH GOPTIONS statement

See: string on page 296

BODYSCROLLBAR=YES | NO | AUTO
specifies whether or not to put a scrollbar in the frame that references the body file.

YES
places a scrollbar in the frame that references the body file.

NO
specifies not to put a scrollbar in the frame that references the body file.

AUTO
places a scrollbar in the frame that references the body file only if needed.

Tip: Typically, BODYSCROLLBAR is set to AUTO.

 Applies to: frame

ODS Destinations: HTML

BODYSIZE=dimension | number % | *
specifies the width of the frame that displays the body file in the HTML frame file. (For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” on page 637.)

dimension
is a nonnegative number. The unit of measure is pixels.

See: dimension on page 294

number %
specifies the width of the frame 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.

 Applies to: frame

ODS Destinations: HTML

BORDERCOLOR=color
specifies the color of the border if the border is just one color.

 Applies to: tables or cells

ODS Destinations: HTML, RTF, PRINTER family

See: color on page 292
BORDERCOLORDARK=\textit{color}

specifies the darker color to use in a border that uses two colors to create a
three-dimensional effect.

\textbf{Interaction:} If you create HTML4 output, then the BORDERCOLORDARK style
attribute is ignored because it is not part of the HTML4 standard. If you want a
color border, then use the BORDERCOLOR= style attribute.

\textbf{Applies to:} tables or cells

\textbf{ODS Destinations:} HTML, RTF, PRINTER family

\textbf{Featured in:} Example 4 on page 361

\textbf{See also:} color on page 292

BORDERCOLORLIGHT=\textit{color}

specifies the lighter color to use in a border that uses two colors to create a
three-dimensional effect.

\textbf{Interaction:} If you create HTML4 output, then the BORDERCOLORLIGHT style
attribute is ignored because it is not part of the HTML4 standard. If you want a
color border, then use the BORDERCOLOR= style attribute.

\textbf{Applies to:} tables or cells

\textbf{ODS Destinations:} HTML, RTF, PRINTER family

\textbf{Featured in:} Example 4 on page 361

\textbf{See:} color on page 292

BORDERWIDTH=\textit{dimension}

specifies the width of the border of the table.

\textbf{Applies to:} tables

\textbf{ODS Destinations:} HTML, RTF, PRINTER family

\textbf{Tip:} Typically, when BORDERWIDTH=0, the ODS destination sets
RULES=NONE (see the discussion about RULES= on page 317) and
FRAME=VOID (see the discussion about FRAME= on page 310).

\textbf{Featured in:} Example 1 on page 342 and Example 3 on page 355

\textbf{See:} dimension on page 294

BOTTOMMARGIN=\textit{dimension}

specifies the bottom margin for the document.

\textbf{Applies to:} document

\textbf{ODS Destinations:} HTML, RTF, PRINTER family

\textbf{See:} dimension on page 294

BULLETS='\textit{string}'

specifies the string to use for bullets in the contents file. ODS uses bullets in the
contents file. \textit{string} can be one of the following:

- circle
- decimal
- disc
- lower-alpha
- lower-roman
- none
- square
- upper-alpha
- upper-roman.
Applies to: contents

ODS Destinations: HTML

See: string on page 296

CELLHEIGHT=dimension | integer%
specifies the height of the cell. If you specify a percent, it represents a percentage of the height of the table. A row of cells will have the height of the highest cell in the row.

dimension
is a nonnegative number, optionally followed by one of the following units of measure.

See: dimension on page 294

integer%
specifies the height of the cell as a percentage of the height of the table.

Alias: OUTPUTHEIGHT=

Tip: HTML automatically sets cell height appropriately. You should seldom need to specify this attribute in the HTML destination.

Applies to: cells

ODS Destinations: HTML, RTF, PDF, PCL, and PS

CELLPADDING=dimension | integer%
specifies the amount of white space on each of the four sides of the text in a cell.

dimension
is a nonnegative number, optionally followed by one of the following units of measure.

See: dimension on page 294

integer%
specifies the amount of white space on each of the four sides of the text in a cell as a percentage of the table.

Applies to: tables

ODS Destinations: HTML, RTF, PRINTER family

Featured in: Example 3 on page 355

CELLSPACING=dimension
specifies the thickness of the spacing between cells.

Applies to: tables

Interaction: If BORDERWIDTH= is nonzero, and if the background color of the cells contrasts with the background color of the table, then the color of the cell spacing is determined by the table's background.

Featured in: Example 1 on page 342 and Example 3 on page 355

See: dimension on page 294

CELLWIDTH=dimension | integer%
specifies the width of the cell. If you specify a percent, it represents a percentage of the width of the table. A column of cells will have the width of the widest cell in the column.

dimension
is a nonnegative number, optionally followed by one of the following units of measure.

See: dimension on page 294
integer% specifies the width of the cell as a percentage of the width of the table.

**Alias:** OUTPUTWIDTH=

**Applies to:** cells

**ODS Destinations:** HTML, RTF, PRINTER family

CONTENTPOSITION= LEFT | RIGHT | TOP | BOTTOM specifies the position, within the frame file, of the frames that display the contents and the page files. (For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” on page 637.)

**LEFT**
places the frames on the left.

**Alias:** L

**RIGHT**
places the frames on the right.

**Alias:** R

**TOP**
places the frames at the top.

**Alias:** T

**BOTTOM**
places the frames at the bottom.

**Alias:** B

**Applies to:** frame

**ODS Destinations:** HTML

CONTENTSCROLLBAR=YES | NO |AUTO specifies whether or not to put a scrollbar in the frames in the frame file that display the contents and the page files. (For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” on page 637.)

**YES**
places a scrollbar in the frames in the frame file that display the contents and the page files.

**NO**
specifies not to put a scrollbar in the frames in the frame file that display the contents and the page files.

**AUTO**

**Tip:** Typically, CONTENTSCROLLBAR= is set to AUTO.

**Applies to:** frame

**ODS Destinations:** HTML

CONTENTSIZE=dimension | number % | *
specifies the width of the frames in the frame file that display the contents and the page files. (For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” on page 637)

**dimension**
is a nonnegative number. The unit of measure is pixels.

**See:** dimension on page 294
**number %**

specifies the width of the frames as a percentage of the entire display.

**Requirement:** number % must be a positive number between 0 and 100.

* specifies to use whatever space is left after displaying the body file as specified by the BODYSIZE= attribute.

**See also:** BODYSIZE= on page 303

**Applies to:** frame

**ODS Destinations:** HTML

**CONTRASTCOLOR=**color

specifies the alternate colors for maps. The alternate colors are applied to the blocks on region areas in block maps.

**Applies to:** graphs

**ODS Destinations:** HTML

**See:** color on page 292

**DROPShadow=** ON | OFF

determines whether drop shadow effect is used with text.

**ON**

specifies that a drop shadow effect is used with text.

**OFF**

specifies that a drop shadow effect is not used with text.

**Applies to:** graphs

**ODS Destinations:** HTML

**ENDCOLOR=**color

indicates the end 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 ENDCOLOR, then the start color will be completely transparent gradationally to the end color.

**Applies to:** graphs

**ODS Destinations:** HTML

**See:** color on page 292

**FILLRULEWIDTH=** dimension

distinguishes a rule of the specified width to be placed into the space around the text (or entire cell if there is no text) where white space would otherwise appear.

**Tip:** If no text is specified, then FILLRULEWIDTH= fills the space around the text with dash marks. For example: –this– or this ——.

**Applies to:** tables

**ODS Destinations:** PDF, PS, and PCL

**See:** dimension on page 294

**FLYOVER='string’**

specifies the text to show in a tool tip for the cell.

**Applies to:** cells

**ODS Destinations:** HTML and PDF

**See:** string on page 296
FONT=font-definition
specifies a font definition to use.

**Tip:** When you specify this attribute for a table, it 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, you must set the attribute for a cell.

**Tip:** If the system does not recognize the font specified, then it will refer to your system's default font. This attribute does not accept concatenated fonts. SAS Graph Styles can only specify one font.

**Applies to:** tables, cells, and graphs

**ODS Destinations:** HTML, RTF, PRINTER family

**Featured in:** Example 3 on page 355

**See:** font definition on page 294

FONT_FACE=’string-1<... string-n>’
specifies the font to use. If you supply multiple fonts, then the destination device uses the first one that is installed on your system.

You cannot be sure what fonts are available to someone who is viewing your output in a browser or printing it on a high-resolution printer. Most devices support

- times
- courier
- arial, helvetica.

**Tip:** When you specify this attribute for a table, it 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, you must set the attribute for a cell.

**Applies to:** cells and graphs

**ODS Destinations:** HTML, RTF, PRINTER family

**Featured in:** Example 1 on page 342

**See:** string on page 296

FONT_SIZE=dimension | size
specifies the size of the font.

**dimension**
is a nonnegative number.

**See:** dimension on page 294

**Restriction:** If you specify a dimension, then you must specify a unit of measure. Without a unit of measure, the number becomes a relative size.

**size**
The value of size is relative to all other font sizes in the document.

**Range:** 1 to 7, for size

**Tip:** When you specify this attribute for a table, it 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, you must set the attribute for a cell.

**Applies to:** table, cells, and graphs

**ODS Destinations:** HTML, RTF, PDF, PCL, and PS

**Featured in:** Example 1 on page 342
FONT_STYLE=ITALIC | ROMAN | SLANT
specifies the style of the font. In many cases, italic and slant map to the same font.

**Tip:** When you specify this attribute for a table, it 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, you must set the attribute for a cell.

**Applies to:** tables, cells, and graphs

**ODS Destinations:** HTML, PCL, PDF, PS, and RTF

**Featured in:** Example 1 on page 342 and Example 3 on page 355

FONT_WEIGHT=weight
specifies the font weight. *weight* can be any of the following:

- MEDIUM
- BOLD
- DEMI_BOLD
- EXTRA_BOLD
- LIGHT
- DEMI_LIGHT
- EXTRA_LIGHT.

**Restriction:** You cannot be sure what font weights are available to someone who is viewing your output in a browser or printing it on a high-resolution printer. Most devices support only MEDIUM and BOLD, and possibly LIGHT.

**Tip:** When you specify this attribute for a table, it 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, you must set the attribute for a cell.

**Applies to:** tables, cells, and graphs

**ODS Destinations:** HTML, PCL, PDF, PS, and RTF

**Featured in:** Example 1 on page 342

FONT_WIDTH=relative-width
specifies the font width compared to the width of the usual design. *relative-width* can be any of the following:

- NORMAL
- COMPRESSED
- EXTRA_COMPRESSED
- NARROW
- WIDE
- EXPANDED.

**Restriction:** Few fonts honor these values.

**Tip:** When you specify this attribute for a table, it 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, you must set the attribute for a cell.

**Applies to:** tables, cells, and graphs

**ODS Destinations:** HTML, RTF, PCL, PDF, PS, and RTF

**Featured in:** Example 1 on page 342
FOREGROUND=color
specifies the color of the foreground, which is primarily the color of text.

Tip: When you specify this attribute for a table, it 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, you must set the attribute for a cell.

Applies to: tables, cells, and graphs

ODS Destinations: HTML, PCL, PDF, PS, and RTF

Overridden by: CBACK= option in the SAS/GRAPH GOPTIONS statement

Featured in: Example 3 on page 355

See: color on page 292

FRAME=frame-type
specifies the type of frame to use on a table. The following 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>

Applies to: tables

ODS Destinations: HTML, PRINTER family, and RTF

Featured in: Example 3 on page 355

FRAMEBORDER=ON | OFF
specifies whether or not 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.

Applies to: frame

ODS Destinations: HTML

FRAMEBORDERWIDTH=dimension
specifies the width of the border around the frames for an HTML file that uses frames.

Applies to: frame

ODS Destinations: HTML
See: *dimension on page 294*

**FRAMESPACING=integer**  
specifies the width of the space between frames for HTML that uses frames.  
**Applies to:** frame  
**ODS Destinations:** HTML

**GRADIENT_DIRECTION= XAXIS | YAXIS**  
specifies the direction for the gradient effect for a graph's background, legend background, charts, walls, and floors. Use XAXIS for a left-to-right gradient and YAXIS for a bottom-to-top gradient.  
**Applies to:** graphs  
**ODS Destinations:** HTML

**HREFTARGET=target**  
specifies the window or frame in which to open the target of the link. *target* can be one of the following values.  
- **_BLANK**  
  opens the target in a new, blank window. The window has no name.  
- **_PARENT**  
  opens the target in the window from which the current window was opened.  
- **_SEARCH**  
  opens the target in the browser's search pane.  
  **Restriction:** Only available in Internet Explorer 5.0 or later.  
- **_SELF**  
  opens the target in the current window.  
- **_TOP**  
  opens the target in the topmost window.  
- **'name'**  
  opens the target in the specified window or the frame.  
**Default:** _SELF  
**Applies to:** cells  
**ODS Destinations:** HTML

**HTMLCLASS='string'**  
specifies the name of the style sheet class to use for the table or cell.  
**Applies to:** document  
**ODS Destinations:** HTML  
**See:** *string on page 296*

**HTMLCONTENTTYPE='string’**  
provides the value of the content type for pages that you send directly to a web server rather than to a file.  
**Tip:** The value of *string* is usually “text/html”.  
**Applies to:** document  
**ODS Destinations:** HTML  
**See:** *string on page 296*

**HTMLDOCTYPE='string’**  
specifies the entire doctype declaration for the HTML document, including the opening “<!DOCTYPE” and the closing “>”.
Applies to: document
ODS Destinations: HTML
See: string on page 296

HTMLID='string'
specifies an id for the table or cell. The id is for use by a Javascript.
Applies to: tables and cells
ODS Destinations: HTML
See: string on page 296

HTMLSTYLE='string'
specifies individual attributes and values for the table or cell.
Applies to: document
ODS Destinations: HTML
See: string on page 296

IMAGE=string
specifies the image to appear in the graph. This image can be positioned or tiled.
Applies to: graphs
ODS Destinations: HTML

Overridden by: IBACK= and IMAGESTYLE=TILE options in the SAS/GRAPH GOPTIONS statement
See: string on page 296

INDENT=n
specifies that the output be indented one more indentation level, using the number of spaces specified by the INDENT= statement.
Default: The default value for XML is 2. For all other ODS destinations, the default value is 0.
ODS Destinations: MARKUP, RTF and PRINTER family

n
specifies the number of spaces that you want the output to indent.

JUST=CENTEIR | DEC | LEFT | RIGHT
specifies justification. In graphs, this option specifies the justification of the image specified with the IMAGE= statement.

CENTER
specifies center justification.
Alias: C

DEC
specifies aligning the values by the decimal point.
Alias: D

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.
**Interaction:** If the column is numeric, then values are aligned to the right if you specify JUST=C and JUSTIFY=OFF.

**Interaction:** All destinations except LISTING justify the values in columns as if JUSTIFY=ON for JUST=R and JUST=L.

**Main discussion:** “How Are Values in Table Columns Justified?” on page 512

**Applied to:** tables, cells, and graphs

**ODS Destinations:** HTML, PCL, PDF, PS, and RTF

**Tip:** For Printer Family destinations and the MARKUP destination, you can use the style attribute JUST= with the style attribute VJUST= in the style element PAGENO to control the placement of page numbers.

For example, the following statement would produce a page number that is centered at the bottom of the page:

```
style PageNo from TitleAndFooters / just=c vjust=b;
```

**Tip:** For Printer Family destinations and the MARKUP destination, you can control the placement of dates by using the style attribute JUST= with the style attribute VJUST= in any of the following style elements:

- BODYDATE
- DATE.

For example, the following statement would produce a date in the body file that is left justified at the top of the page:

```
style BodyDate from Date / just=l vjust=t;
```

**LEFTMARGIN=**

specifies the left margin for the document.

**Applies to:** document

**ODS Destinations:** HTML, RTF, PRINTER family

**See:** dimension on page 294

**LINESTYLE= 1...46**

controls the line style for a graph. Possible values are SAS/GRAPH line types one through 46. If LINESTYLE=1, then a solid line is drawn. Dashed lines are drawn when values between (and including) two and 46 are specified as the LINESTYLE= value.

**Applies to:** graphs

**ODS Destinations:** HTML, RTF, PRINTER family

**See also:** SAS/GRAPH Reference, Volumes 1 and 2

**LINETHICKNESS=**

dimension | number%

specifies the thickness (width) of a line that is part of a graph. This attribute may appear in many style elements that pertain to graphs such as GraphAxisLines and GraphBorderLines. If you specify a percent, it represents a percentage of the width of the window or display.

**dimension**

is a nonnegative number.

**See:** dimension on page 294

**number%**

**Restriction:** The LINETHICKNESS= attribute does not apply to output generated as a result of GRSEG (graph segment) output.

**Overridden by:** WIDTH= option in the AXIS or SYMBOL statement, or other options that are specific to charts which set line width.


**Applies to:** graphs  
**ODS Destinations:** HTML, RTF, Printer Family  
**See also:** SAS/GRAPH Reference, Volumes 1 and 2

**LINKCOLOR=**color  
specifies the color for links that have not yet been visited.  
**Applies to:** document  
**ODS Destinations:** HTML, RTF, and PDF  
**See:** color on page 292

**LISTENTRYANCHOR=**ON | OFF  
specifies whether or not to make this entry in the table of contents a link to the body file.  
**Applies to:** document  
**ODS Destinations:** HTML

**LISTENTRYDBLSpace=**ON | OFF  
specifies whether or not to double space between entries in the table of contents.  
**Applies to:** document  
**ODS Destinations:** HTML

**NOBREAKSPACE=**ON | OFF  
specifies how to handle space characters.  
**Applies to:** cells  
**ODS Destinations:** All

**OUTPUTHEIGHT=**dimension  
specifies the height for a graph or graphics in a document.  

*Note:* When used with graphs, the OUTPUTHEIGHT=**dimension** must be specified as a pixel or percentage value. If a unit of measure is not specified with the **dimension**, then the value will be in pixels. If a unit of measure other than pixels or percentage is specified with the **dimension**, then the OUTPUTHEIGHT=**dimension** is not applied to the graph.  

**Alias:** CELLHEIGHT=  
**Restriction:** The OUTPUTHEIGHT=**dimension** does not apply to output generated as a result of GRSEG (graph segment) output.  
**Applies to:** graphs and documents  
**ODS Destinations:** HTML, RTF, PRINTER family
**Overridden by:** YPIXELS= option in the SAS/GRAPH GOPTIONS statement

**See:** dimension on page 294

**OUTPUTWIDTH=** \textit{dimension} | \textit{number\%}

specifies the width of a table, line, or a graph. If you specify a percent, it represents a percentage of the width of the window or display.

\textit{Note:} When used with graphs, the OUTPUTHEIGHT=\textit{dimension} 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 OUTPUTHEIGHT=\textit{dimension} is not applied to the graph. △

dimension

is a nonnegative number.

**See:** dimension on page 294

\textit{number\%}

**Alias:** CELLWIDTH=

**Restriction:** The OUTPUTHEIGHT= option does not apply to output generated as a result of GRSEG (graph segment) output.

**Tip:** Use OUTPUTWIDTH=100\% to make the table or graph as wide as the window that it is open in.

**Applies to:** tables and graphs

**ODS Destinations:** HTML, RTF, PRINTER family

**Overridden by:** XPIXELS= option in the SAS/GRAPH GOPTIONS statement

**OVERHANGFACTOR=** \textit{nonnegative-number}

specifies an upper limit for extending the width of the column.

**Tip:** Typically, an overhang factor between 1 and 2 works well.

**Tip:** The HTML that is generated by ODS tries to ensure that the text in a column wraps when it reaches the requested column width. If you make the overhang factor greater than 1, then the text can extend beyond the specified width.

**Applies to:** document

**ODS Destinations:** HTML, RTF, PRINTER family

**PAGEBREAKHTML=** `'string'`

specifies HTML to place at page breaks.

**Applies to:** document

**ODS Destinations:** HTML

**See:** string on page 296

**POSTHTML=** `'string'`

specifies the HTML code to place after the table or cell.

**Applies to:** tables or cells

**ODS Destinations:** HTML

**Featured in:** Example 3 on page 355

**See:** string on page 296

**POSTIMAGE=** `'string'` | fileref

specifies an image to place before the table or cell.
string
names a GIF or JPEG file. You can use a simple filename, a complete path, or a
URL.
See: string on page 296

fileref
is a reference that has been assigned to an external file. Use the FILENAME
statement to assign a fileref. (For information about the FILENAME statement,
see “Statements” in SAS Language Reference: Dictionary.)

Applies to: tables or cells
ODS Destinations: HTML, PCL, PDF, PS, and RTF

POSTTEXT='string'
specifies text to place after the cell or table.
Applies to: tables or cells
ODS Destinations: HTML, PCL, PDF, PS, and RTF
See: string on page 296

PREHTML='string'
specifies the HTML code to place before the table or cell.
Applies to: tables or cells
ODS Destinations: HTML
See: string on page 296

PREIMAGE= 'string' | fileref
specifies an image to place before the table or cell.
string
names a GIF or JPEG file. You can use a simple filename, a complete path, or a
URL.
See: string on page 296

fileref
is a reference that has been assigned to an external file. Use the FILENAME
statement to assign a fileref. (For information about the FILENAME statement,
see “Statements” in SAS Language Reference: Dictionary.)

Applies to: tables or cells
ODS Destinations: HTML, PCL, PDF, PS, and RTF

PRETEXT='string'
specifies text to place before the cell or table.
Applies to: tables or cells
ODS Destinations: HTML, PCL, PDF, PS, and RTF
See: string on page 296

PROTECTSPECIALCHARACTERS=ON | OFF | AUTO
determines how less-than signs (<), greater-than signs (>), and ampersands (&)
are interpreted. 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 your 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.

 Applies to: cells

ODS Destinations:  HTML, MARKUP family, PRINTER family, and RTF

RIGHTMARGIN=dimension
specifies the right margin for the document.

 Applies to: document

ODS Destinations:  HTML, RTF, PRINTER family

See: dimension on page 294

RULES=rule-type
specifies the types of rules to use in a table. The following table shows the possible values for rule and their meanings:

<table>
<thead>
<tr>
<th>This value of rule ...</th>
<th>Creates rules in these locations</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>

 Applies to: tables

ODS Destinations:  HTML, RTF, PRINTER family

Featured in: Example 4 on page 361

STARTCOLOR= color
indicates the start fill color for a graph. It is used to create a gradient effect.

Note: You can have either a start and end gradient effect or no gradient effect. If you specify a TRANSPARENCY level and you only specify the STARTCOLOR,
then the end color will be completely transparent gradationally to the specified start color.

**Applies to:** graphs

**ODS Destinations:** HTML

**See:** color on page 292

**TAGATTR=’string’**
specifies text to insert in the HTML. The string must be valid HTML for the context in which the style element is created. 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.

**Applies to:** cells

**ODS Destinations:** HTML

**See:** string on page 296

**TOPMARGIN=dimension**
specifies the top margin for the document.

**Applies to:** document

**ODS Destinations:** HTML, RTF, PRINTER family

**See:** dimension on page 294

**TRANSPARENCY=dimension**
specifies a transparency level. Valid values are 0.0 (opaque) to 1.0 (transparent).

**Applies to:** graphs

**ODS Destinations:** HTML

**See:** dimension on page 294

**URL=’uniform-resource-locator’**
specifies a URL to link to from the current cell.

**Applies to:** cells

**ODS Destinations:** HTML, RTF, and PDF

**VISITEDLINKCOLOR=color**
specifies the color for links that have been visited.

**Applies to:** document

**ODS Destinations:** HTML and RTF

**See:** color on page 292

**VJUST=BOTTOM | MIDDLE | TOP**
specifies vertical justification. In graphs, this option specifies the vertical justification of the image specified with IMAGE=.

**BOTTOM**
specifies bottom justification.

**Alias:** B

**MIDDLE**
specifies center justification.

**Alias:** M

**TOP**
specifies top justification.

**Alias:** T

**Applies to:** cells and graphs
ODS Destinations:  HTML, PCL, PDF, PS, and RTF

Tip: For Printer Family destinations and the MARKUP destination, you can use the style attribute VJUST= with the style attribute JUST= in the style element PAGENO to control the placement of page numbers.

For example, the following statement produces a page number that is centered at the bottom of the page:

```
style PageNo from TitleAndFooters / just=c vjust=b;
```

Tip: For Printer Family destinations and the MARKUP destination, you can control the placement of dates by using the style attribute VJUST= with the style attribute JUST= in any of the following style elements:

- BODYDATE
- DATE

For example, the following statement produces a date in the body file that is left justified at the top of the page:

```
style BodyDate from Date / just=l vjust=t;
```

WATERMARK=ON | OFF

specifies whether or not to make the image that is specified by BACKGROUNDIMAGE= into a “watermark.” A watermark appears in a fixed position as the window is scrolled.

ON

specifies to make the image that is specified by BACKGROUNDIMAGE= into a “watermark.”

OFF

specifies not to make the image that is specified by BACKGROUNDIMAGE= into a “watermark.”

Applies to:  document

ODS Destinations:  HTML

---

**END Statement**

Ends the style definition

```
END;
```

---

**Concepts: Style Definitions and the TEMPLATE Procedure**

**Viewing the Contents of a Style Definition**

To view the contents of a style definition, you can use the SAS windowing environment, the command line, or the TEMPLATE procedure.
The Default Style Definition for HTML and Markup Languages

Where Is the Default Style Definition for HTML and Markup Languages?

The default style definition for the HTML and markup languages destinations are stored in STYLES.DEFAULT in the template store SASHELP.TMPLMST. You can view the style definition from the TEMPLATE window, or you can submit this PROC TEMPLATE step to write the style definition to the SAS log:

```sas
proc template;
  source styles.default;
run;
```
Modifying Style Elements in the Default Style Definition for HTML and Markup Languages

When you are working with style definitions, you are more likely to modify a SAS style definition than to write a completely new style definition. Example 3 on page 355 shows you how to modify the default style definition.

When you want to customize the style definition for use at your site, it is helpful to know what each style element in the style definition specifies. For a list of the default HTML and markup languages style elements, see Appendix 4, “HTML, Printer Family, and Markup Languages Style Elements and Their Inheritances,” on page 651.

Note: The default style definition for the PRINTER destination is stored in STYLES.PRINTER in the template store SASHELP.TMPLMST. Similarly, the default style definition for the RTF destination is stored in STYLES.RTF in the template store SASHELP.TMPLMST.

ODS Styles with Graphical Style Information

SAS provides ODS styles that incorporate graph style information. See “Viewing the Contents of a Style Definition” on page 319 for information about viewing the code for the ODS styles delivered with SAS. In addition to using defined ODS styles, you can also modify an existing style or create an entirely new style using the new graph style elements. Example 4 on page 361 describes how a defined ODS style was generated. See “Style Attributes and Their Values” on page 292 for a complete listing of style attributes. For a complete list of style elements see Appendix 4, “HTML, Printer Family, and Markup Languages Style Elements and Their Inheritances,” on page 651.

Note: The graph styles (attributes and elements) are at the bottom of the style attributes and style elements tables.

While graph styles utilize a number of attributes that are also used by other styles generated with PROC TEMPLATE, several attributes are unique to graph styles. For example, you can use STARTCOLOR and ENDCOLOR to produce a gradient effect that gradually changes from the starting color to the ending color in a specified element. When only either a STARTCOLOR or ENDCOLOR, but not both, is specified, then the attribute that was not specified is transparent when TRANSPARENCY is being used. In Example 4 on page 361, only an ENDCOLOR is specified; therefore, the starting color is transparent.

TRANSPARENCY is another attribute unique to graph styles. With transparency, you can 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 BACKGROUNDIMAGE in other styles to stretch an image, in graph styles you can also use IMAGE to position or tile an image.

With graph styles you can also combine images and colors to create a blending affect. The blending works best when you use a grayscale image with a specified color. Blending can be done in the following elements: GraphLegendBackground, GraphCharts, GraphData#, GraphFloor, and GraphWalls. To blend, specify a color using the BACKGROUND or FOREGROUND attribute and specify an image using the BACKGROUNDIMAGE or IMAGE attribute.

Note: When using the GraphData# element, you can use the FOREGROUND attribute, but not the BACKGROUND attribute to specify a color value.
About Style Definition Inheritance and Style Element Inheritance

Definitions

To help you become familiar with style definition inheritance and style element inheritance, let's review the definitions of a style definition and a style element.

**style definition** describes how to display the presentation aspects (color, font, font size, and so on) of the output for an entire SAS job. A style definition determines the overall appearance of the documents that use it. Each style definition is composed of style elements.

**style element** is a collection of style attributes that apply to a particular part of the output for a SAS job. For example, a style element may contain instructions for the presentation of column headers or for the presentation of the data inside cells. Style elements may also specify default colors and fonts for output that uses the style definition. 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 FONT_STYLE= attribute specifies whether to use a Roman, a slant, or an italic font.

When you use PROC TEMPLATE to create style definitions, it is important to understand inheritance. There are two types of inheritance:

**style definition inheritance** specifies that the child style definition receives all of the style elements and attributes and statements that are specified in the parent's definition. They are used in the new definition unless the new definition overrides them.

**style element inheritance** specifies that the child style element receives all of the elements and their attributes that are specified in another style definition. They are used in the new style definition unless the new definition overrides them. Each style attribute specifies a value for one aspect of the presentation. For example, a style element may contain instructions for the presentation of column headers or for the presentation of the data inside cells. Style elements may also specify default colors and fonts for output that uses the style definition.

**Note:** For a list of the default style elements used for HTML and markup languages and their inheritance, see Appendix 4, “HTML, Printer Family, and Markup Languages Style Elements and Their Inheritances,” on page 651.
How to Determine Style Definition Inheritance

A style definition determines the overall appearance of the documents that use it. Each style definition is composed of style elements. A style definition is created with the DEFINE STYLE statement and its substatements and attributes.

The PARENT= attribute, used with the DEFINE STYLE statement, determines style definition inheritance. When you specify a parent for a style definition, all the style elements, attributes, and statements that are specified in the parent's definition are used in the new definition unless the new definition overrides them.

How to Determine Style Element Inheritance

The STYLE and REPLACE statements, used with the DEFINE STYLE statement, determine style element inheritance. They augment or override the attributes of a particular style element. You can use the STYLE statement in either a style definition that has no parent or a style definition that has a parent. However, you can use the REPLACE statement only in a style definition that has a parent.

Creating a Style Definition with No Parent, Using Style Element Inheritance

This section explains style definition inheritance and style element inheritance, beginning with the simpler case of style element inheritance in a style definition that has no parent and progressing to more complicated cases. The focus here is on PROC TEMPLATE and the DEFINE STYLE statement, so only the PROC TEMPLATE code that creates the style definitions appears in the text. However, in order to produce the HTML output that is shown here, it is necessary to create a customized table and to bind that table to a data set. The complete code that produces each piece of output is in “Programs that Illustrate Inheritance” on page 622.

Creating a Style Element in a Style Definition

Use a DEFINE STYLE statement to create each style element in the style definition. The following PROC TEMPLATE step creates the style definition, `concepts.style1`, that contains one style element, `celldatasimple`.

### Example Code 9.1  Creating a Style Definition with One Style Element

```
proc template;
    define style concepts.style1;
    style celldatasimple /
        font_face=arial
        background=very light vivid blue
        foreground=white;
    end;
run;
```

This style element contains the following style attributes:

- Arial font
- light blue background
- white foreground.

The style element `celldatasimple` does not inherit any attributes from any other element. It is simply created with the three attributes shown. All other attributes are set by the browser when a table is displayed in HTML with this style definition. The following HTML output uses the following style definition.
Display 9.3  Using a Style Definition with One Style Element

The style definition for this HTML output uses contains only one style element: `celldatasimple`. All three columns use this style element. `celldatasimple` contains the following style attributes:
- `FONT_FACE=arial`
- `BACKGROUND=very light vivid blue`
- `FOREGROUND=white`.

Creating a Second Style Element in a Style Definition

You can create a second style element in a style definition either independently of any other style element or from an existing style element.

Suppose that you want an additional style element that emphasizes the data for cells by using an italic font. The style element uses the same font and background color as `celldatasimple`, but it uses blue instead of white for the foreground color. Program 1 shows you how you can create the new style element independently. Alternatively, you can create `celldataemphasis` from `celldatasimple` as shown in program 2.

Example Code 9.2  Program 1: Creating a Second Style Element Independently or from an Existing Style Element

The PROC TEMPLATE steps in the following code produce identical results. In both cases, `celldatasimple` contains the following style attributes:
- `FONT_FACE=arial`
- `BACKGROUND=very light vivid blue`
- `FOREGROUND=white`.

`celldataemphasis` contains the following style attributes:
- `FONT_FACE=arial` (inherited from `celldatasimple`)
- `BACKGROUND=very light vivid blue` (inherited from `celldatasimple`)
- `FOREGROUND=blue` (modified in `celldataemphasis`)
- `FONTSTYLE=italic` (added in `celldataemphasis`).

```plaintext
proc template;
  define style concepts.style1;
```
style celldatasimple /
  font_face=arial
  background=very light vivid blue
  foreground=white;
style celldataemphasis /
  font_face=arial
  background=very light vivid blue
  foreground=blue
  font_style=italic;
end;
run;

Example Code 9.3  Program 2: Creating a Second Style Element from an Existing Style Element

proc template;
  define style concepts.style1;
    style celldatasimple /
      font_face=arial
      background=very light vivid blue
      foreground=white;
    style celldataemphasis from celldatasimple /
      foreground=blue
      font_style=italic;
  end;
run;

The following HTML output uses the style definition concepts.style1.

Display 9.4  A Style Definition with Two Style Elements

The style definition that this HTML output contains two style elements: celldatasimple and celldataemphasis. The columns for Country and Kilotons use celldatasimple. The column for grain uses celldataemphasis.
Comparing the Two Methods

Although the two PROC TEMPLATE steps above produce identical HTML output, there is an important difference between them that is illustrated by programs 3 and 4.

Program 3 below does not use style element inheritance. `celldatasimple` is created independently of `celldatasimple`, so a change to `celldatasimple` does not affect `celldatasimple`. Even if you change the STYLE statement that creates `celldatasimple` so that the font is Times, then the program still creates `celldatasimple` with Arial as the font.

However, in program 4, if you change the font for `celldatasimple` from Arial to Times, then `celldatasimple` does use the Times font. This is because the change to FONT_FACE= is passed to `celldatasimple`, which inherits all the attributes from `celldatasimple`.

Example Code 9.4  Program 3: Changing the Font in Only One Style Element

```plaintext
proc template;
define style concepts.style1;
style celldatasimple /
  font_face=times
  background=very light vivid blue
  foreground=white;
style celldatasimpleemphasis /
  font_face=arial
  background=very light vivid blue
  foreground=blue
  font_style=italic;
end;
```

Example Code 9.5  Program 4: Changing the Font in the Parent and Child Style Elements

```plaintext
proc template;
define style concepts.style1;
style celldatasimple /
  font_face=times
  background=very light vivid blue
  foreground=white;
style celldatasimpleemphasis from celldatasimple /
  foreground=blue
  font_style=italic;
end;
```

Output For Comparing The Two Methods

The following HTML output uses the style definition created by program 3.
Display 9.5  Changing the Font in Only One Style Element

Here, the font in the style element `celldatasimple`, which is used for the first and third columns in the HTML output, has changed from Arial to Times. However, `celldataemphasis`, which is used for the second column, still uses the Arial font because it does not inherit any attributes from `celldatasimple`.

The following HTML output uses this style definition created by program ④.

Display 9.6  Inheriting a Change to a Style Element

In this case, the change to the Times font in `celldatasimple` is inherited by `celldataemphasis`. Both style elements use the Times font. The only attributes that differ between the two style elements are attributes that were explicitly redefined in the definition of `celldataemphasis` (the FOREGROUND= attribute, which was changed, and the FONT_STYLE= attribute, which was added). The columns for Country and Kilotons use `celldatasimple`. The column for Grain uses `celldataemphasis`.

Adding a Third Style Element

In this example, a third style element is added to the style definition. This style element further emphasizes the data by using a large, bold, italic font. Again, you
can create the new style element from scratch, or you can derive it from either of the other style elements. The following program creates `celldatalarge` from `celldataemphasis`:

**Example Code 9.6** Program 5: Creating the Style Element `celldatalarge`

```sas
proc template;
define style concepts.style1;
  style celldatasimple /
    font_face=arial
    background=very light vivid blue
    foreground=white;
  style celldataemphasis from celldatasimple /
    foreground=blue
    font_style=italic;
  style celldatalarge from celldataemphasis /
    font_weight=bold
    font_size=5;
end;
run;
```

The style elements `celldatasimple` and `celldataemphasis` have not changed. `celldatasimple` has these attributes:
- FONT_FACE=arial
- BACKGROUND=very light vivid blue
- FOREGROUND=white.

`celldataemphasis` has these attributes:
- FONT_FACE=arial (inherited from `celldatasimple`)
- BACKGROUND=very light vivid blue (inherited from `celldatasimple`)
- FOREGROUND=blue (modified in `celldataemphasis`)
- FONT_STYLE=italic (added in `celldataemphasis`).

The new style element, `celldatalarge`, has these attributes:
- FONT_FACE=arial (inherited from `celldataemphasis`, which inherited it from `celldatasimple`)
- BACKGROUND=very light vivid blue (inherited from `celldataemphasis`, which inherited it from `celldatasimple`)
- FOREGROUND=blue (inherited from `celldataemphasis`)
- FONT_STYLE=italic (inherited from `celldataemphasis`)
- FONT_WEIGHT=bold (added in `celldatalarge`)
- FONT_SIZE=5 (added in `celldatalarge`).

The following HTML output uses the new style definition created by program 5.
Display 9.7  Adding the Style Element celldatalarge

The style definition that this HTML output uses contains three style elements: `celldatasimple`, `celldataemphasis`, and `celldatalarge`. The column for Country uses `celldatasimple`. The column for Grain uses `celldataemphasis`. The column for Kilotons uses `celldatalarge`.

In this case, `celldatalarge` inherits style attributes from `celldataemphasis`, and `celldataemphasis` inherits from `celldatasimple`. If you change the font in `celldatasimple`, then the font in the other style elements also changes. If you change the font style or foreground color in `celldataemphasis`, then the font style or foreground color in `celldatalarge` also changes. Changes to `celldatalarge` affect only `celldatalarge` because no style element inherits from it.

Summary of Style Element Inheritance in a Style Definition with No Parent

The following points summarize style element inheritance in a style definition that does not have a parent:

- You can create a new style element from any existing style element.
- The new style element inherits all the attributes from its parent.
- You can specify additional attributes in the new style definition. The attributes are added to the attributes that the element inherits.
- You can change the value of an inherited attribute by respecifying it in the definition of the new style element.

Creating a Style Definition with a Parent Using Style Element Inheritance

- Using One Style Definition to Create Another Style Definition

  Use the PARENT= attribute in a new style definition to inherit an entire style definition.

  This example uses `concepts.style1`, which was created in “Creating a Style Definition with No Parent, Using Style Element Inheritance” on page 323. The following program creates a new style definition, `concepts.style2`, which
inherits the entire style definition from its parent, concepts.style1. At this point, the two style definitions are identical:

**Example Code 9.7** Using Style Definition Inheritance to Create a New Style Definition

```sas
proc template;
   define style concepts.style1;
      style celldatasimple /
         font_face=arial
         background=very light vivid blue
         foreground=white;
      style celldataemphasis from celldatasimple /
         foreground=blue
         font_style=italic;
      style celldatalarge from celldataemphasis /
         font_weight=bold
         font_size=5;
   end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
   end;
run;
```

You can change the new style definition either independently of any other style definitions with a parent by simply overriding the style elements, or by using a style element from another parent style definition.

### Creating a Style Element in a Style Definition with a Parent

**Creating a New Style Definition with a Parent**

You can control the style definition inheritance with the PARENT= attribute of the DEFINE STYLE statement. When you specify a parent for a style definition, all the style elements, attributes, and statements that are specified in the parent’s definition are used in the new definition unless the new definition overrides them.

In this example, a new style element is added to concepts.style2. The following program adds celldatasmall, a style element that does not exist in the parent style definition. Its definition is not based on any other style element.

**Example Code 9.8** Creating a Style Element Independently in a Style Definition with a Parent

```sas
proc template;
   define style concepts.style1;
      style celldatasimple /
         font_face=arial
         background=very light vivid blue
         foreground=white;
      style celldataemphasis from celldatasimple /
         foreground=blue
         font_style=italic;
      style celldatalarge from celldataemphasis /
         font_weight=bold
         font_size=5;
   end;
run;
```
proc template;
  define style concepts.style2;
  parent=concepts.style1;
  style celldatasmall /
    font_face=arial
    background=very light vivid blue
    foreground=blue
    font_style=italic
    font_weight=bold
    font_size=2;
  end;
run;

proc template;
  define style concepts.style2;
  parent=concepts.style1;
  style celldatasmall from celldatalarge /
    font_size=2;
  end;
run;

If you look at the attributes for celldatasmall, you can see that they match the attributes for celldatalarge in the parent style definition, except for FONT_SIZE=.

Creating a Style Element from a Style Element in a Parent Definition

Another way to create this new style element, is to create it from celldatalarge. You do this just as you did when you created a style element in a style definition that did not have a parent:

Example Code 9.9 Creating a New Style Element from a Style Element in the Parent Style Definition

proc template;
  define style concepts.style1;
  style celldatasimple /
    font_face=arial
    background=very light vivid blue
    foreground=white;
  style celldataemphasis from celldatasimple /
    foreground=blue
    font_style=italic;
  style celldatalarge from celldataemphasis /
    font_weight=bold
    font_size=5;
  end;
run;

proc template;
  define style concepts.style2;
  parent=concepts.style1;
  style celldatasmall from celldatalarge /
    font_size=2;
  end;
run;

When you specify the FROM option in the STYLE statement of a style definition with a parent, PROC TEMPLATE first searches in the child style definition for the style element that you specify. If no such style element exists, it searches in the parent style definition and continues searching up through the hierarchy of parents. In this case, because no style element called celldatalarge exists in concepts.style2, PROC TEMPLATE uses the style element from the parent style definition.
Comparing the Style Element celldatasmall

The style definition concepts.style2 that is produced in program 6 below is identical to the one that is produced in program 7. In both cases, the style element celldatasmall has these attributes:

- FONT_FACE=arial (inherited from celldatalarge through celldataemphasis and celldatasimple)
- BACKGROUND=very light vivid blue (inherited from celldatalarge through celldataemphasis and celldatasimple)
- FOREGROUND=blue (inherited from celldatalarge through celldataemphasis)
- FONT_STYLE=italic (inherited from celldatalarge through celldataemphasis)
- FONT_WEIGHT=bold (inherited from celldatalarge)
- FONT_SIZE=2 (modified in celldatasmall).

Example Code 9.10  Program 6: Creating a Style Element Independently in a Style Definition with a Parent

```zubrinam
proc template;
define style concepts.style1;
  style celldatasimple/
    font_face=arial
    background=very light vivid blue
    foreground=white;
  style celldataemphasis from celldatasimple /
    foreground=blue
    font_style=italic;
  style celldatalarge from celldataemphasis /
    font_weight=bold
    font_size=5;
end;
run;

proc template;
define style concepts.style2;
  parent=concepts.style1;
  style celldatasmall /
    font_face=arial
    background=very light vivid blue
    foreground=blue
    font_style=italic
    font_weight=bold
    font_size=2; end;
run;
```

Example Code 9.11  Program 7: Creating a New Style Element from a Style Element in the Parent Style Definition

```zubrinam
proc template;
define style concepts.style1;
  style celldatasimple /
    font_face=arial
    background=very light vivid blue
    foreground=white;
```
style celldataemphasis from celldatasimple / 
   foreground=blue 
   font_style=italic;
style celldatalarge from celldataemphasis / 
   font_weight=bold 
   font_size=5;
end;
run;

proc template;
   define style concepts.style2;
      parent=concepts.style1;
      style celldatasmall from celldatalarge /
         font_size=2;
   end;
run;

The following HTML output uses the style definition concepts.style2.

Display 9.8  Creating a New Style Element from a Style Element in the Parent Style Definition

The style definition concepts.style2 contains four style elements. The style definition inherits celldatasimple, celldataemphasis, and celldatalarge from the parent style definition, concepts.style1. The column for Country uses celldatasimple. The column for grain uses celldataemphasis. The first column for Kilotons uses celldatalarge. The fourth style element in the new style definition is celldatasmall. This style element is created in concepts.style2. It inherits from celldatalarge in concepts.style1. The fourth column, which repeats the values for Kilotons, uses celldatasmall.

<table>
<thead>
<tr>
<th>Country</th>
<th>Grain</th>
<th>Kilotons</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10035</td>
<td>10035</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190100</td>
<td>190100</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>120012</td>
<td>120012</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>51165</td>
<td>51165</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7771</td>
<td>7771</td>
</tr>
</tbody>
</table>

Although program 6 and program 7 above produce the same style definition for concepts.style2, they will produce different style definitions if you change the definition of celldatalarge in the parent (or the definition of any of the style elements that celldatalarge inherits from). In program 8, changes to celldatalarge do not affect celldatasmall because celldatasmall is created
independently in the new style definition. It does not inherit from any style element in the parent style definition.

However, in program 7, changes that you make to `celldatalarge` in the parent style definition do affect `celldatasmall` because `celldatasmall` inherits (and adds to) the attributes of `celldatalarge`. Similarly, changes to other style elements in the parent style definition do not affect `celldatasmall` in program 5, but they do affect `celldatasmall` in program 7.

For example, program 6 below is based on Creating a New Style Element from a Style Element in the Parent Style Definition on page 331. It changes the font in `celldatasimple` from Arial to Times. All the other style elements, in both the parent and the child style definitions, inherit this change. The program also changes the foreground color of `celldataemphasis` to black. The style elements `celldatalarge` (in the parent style definition) and `celldatasmall` (in the new style definition) both inherit this foreground color.

Example Code 9.12  Program 8: Inheriting Changes from Style Elements in the Parent Style Definition

```plaintext
proc template;
define style concepts.style1;
  style celldatasimple /
    font_face=times
    background=very light vivid blue
    foreground=white;
  style celldataemphasis from celldatasimple /
    foreground=black
    font_style=italic;
  style celldatalarge from celldataemphasis /
    font_weight=bold
    font_size=5;
end;
run;

proc template;
define style concepts.style2;
  parent=concepts.style1;
  style celldatasmall from celldatalarge /
    font_size=2;
end;
run;
```

The following HTML output uses the new style definition created by program 6.
Display 9.9 Inheriting Changes to the Parent Style Definition

Changes to the style elements in the parent style definition are passed to style elements that inherit from these elements in both the parent and the child style definitions.

Creating a new style element in a style definition that has a parent is not very different from creating a new style element in a style definition that does not have a parent. The only difference is that the style element that you specify with FROM in the STYLE statement can be in either the parent or the child style definition.

Modifying Existing Style Elements with a Parent

When you create a new style definition from a parent style definition you can, in addition to adding new style elements, modify existing style elements. There are two ways to do this:

- change only the style element that you specify by using the STYLE statement
- change the style element that you specify and all the style elements that inherit from that element by using the REPLACE statement.

The following programs show the results of these methods.

- **Modifying a Style Element by Using the STYLE Statement**

  The following program uses the STYLe statement to redefine the style element `celldatal emphasis` in `concepts.style2`. It changes the background color to white:

  Example Code 9.13 Redefining a Style Element with the STYLE Statement

```plaintext
proc template;
define style concepts.style1;
  style celldatasimple /
    font_face=arial
    background=very light vivid blue
    foreground=white;
  style celldatal emphasis from celldatasimple /
```
ABOUT STYLE DEFINITION INHERITANCE AND STYLE ELEMENT INHERITANCE

Chapter 9

foreground=blue
font_style=italic;
style celldatalarge from celldataemphasis /
  font_weight=bold
  font_size=5;
end;
run;
proc template:
define style concepts.style2;
  parent=concepts.style1;
  style celldataemphasis from celldataemphasis /
    background=white;
  style celldatasmall from celldatalarge /
    font_size=2;
end;
run;

In this case, celldataemphasis in concepts.style2 initially inherits all the attributes of celldataemphasis in concepts.style1 because it is created from this style element. The inherited attributes are

- FONT_FACE=Arial (which celldataemphasis inherits from celldatasimple)
- BACKGROUND= very light vivid blue, (which celldataemphasis inherits from celldatasimple)
- FOREGROUND=white
- FONT_STYLE=italic.

The STYLE statement that creates celldataemphasis in concepts.style2 changes the background color to white. The background color is the only difference between the celldataemphasis style elements in concepts.style2 and concepts.style1.

But, what about celldatalarge in concepts.style2? The celldatalarge style element is not redefined in concepts.style2. It is defined only in concepts.style1, where it inherits all the attributes of celldataemphasis. So the question is, from which style definition is celldataemphasis inherited—from the parent style definition (concepts.style1), or from the child style definition (concepts.style2)? Is the white background inherited or not?

The answer is that the white background is not inherited because the STYLE statement that creates celldataemphasis in the concepts.style2 affects only those style elements that inherit from celldataemphasis and that are defined in the new style definition. Because celldatalarge is defined only in concepts.style1, it does not inherit the changes that are specified in concepts.style2. Similarly, celldatasmall does not inherit the white background because it inherits the background from celldatalarge. The following HTML output uses this modified version of concepts.style2:
Display 9.10  Using the STYLE Statement to Alter an Existing Style Element in the Child Style Definition

A style element that is defined with the STYLE statement in the child style definition does not pass its attributes to style elements that inherit from the like-named style element in the parent style definition. In this case, the change of the background color for `celldataemphasis` is made in the child style definition. The new background color is not inherited by `celldatalarge` because although the background is inherited from `celldataemphasis`, the background is defined in the parent style definition, not the child definition. Nor is the change inherited by `celldatasmall`, which inherits all of its attributes from `celldatalarge` and from the parents of `celldatalarge`, which include `celldataemphasis` (as defined in the parent style definition) and `celldatasimple`.

![Table showing data on country, grain, and kilotons](image)

Now suppose that you want to pass the white background from `celldataemphasis` on to `celldatalarge` even though it is defined only in `concepts.style1`? You can do this by redefining `celldatalarge` in `concepts.style2` with a STYLE statement. This method works well when you are defining only a few style elements.

Example Code 9.14  Redefining a Style Element without Inheritance

```plaintext
proc template;
    define style concepts.style1;
        style celldatasimple /
            font_face=arial
            background=very light vivid blue
            foreground=white;
        style celldataemphasis from celldatasimple /
            foreground=blue
            font_style=italic;
        style celldatalarge from celldataemphasis /
            font_weight=bold
            font_size=5;
```
PROC TEMPLATE:

define style concepts.style2;
    parent=concepts.style1;
    style celldataemphasis from celldataemphasis /
        background=white;
    style celldatalarge from celldataemphasis /
        font_weight=bold
        font_size=5;
    style celldatasmall from celldatalarge /
        font_size=2;
end;
run;

In this case, when PROC TEMPLATE processes the STYLE statement that creates celldatalarge, it looks for a style element named celldataemphasis to inherit from. Because there is such a style element in concepts.style2, PROC TEMPLATE uses that style element. (If there were no such element in concepts.style2, then PROC TEMPLATE would look for one in concepts.style1 and use that one.) Therefore, celldatalarge inherits the new definition of celldataemphasis, which includes the white background. Similarly celldatasmall, which now inherits from celldatalarge in concepts.style2, inherits the white background.

Example Code 9.15 Redefining a Style Element by Using the REPLACE Statement

PROC TEMPLATE;
    define style concepts.style1;
        style celldatasimple /
            font_face=arial
            background=very light vivid blue
            foreground=white;
        style celldataemphasis from celldatasimple /
            foreground=blue
            font_style=italic;
        style celldatalarge from celldataemphasis /
            font_weight=bold
            font_size=5;
    end;
run;

Example Code 9.16 Redefining a Style Element with the REPLACE Statement

PROC TEMPLATE;
    define style concepts.style2;
        parent=concepts.style1;
        style celldataemphasis from celldataemphasis /
            background=white;
        style celldatalarge from celldataemphasis /
            font_weight=bold
            font_size=5;
        style celldatasmall from celldatalarge /
            font_size=2;
        style celldatalarge from celldataemphasis /
            font_weight=bold
            font_size=5;
    end;
run;

In this case, when PROC TEMPLATE processes the STYLE statement that creates celldatalarge, it looks for a style element named celldataemphasis to inherit from. Because there is such a style element in concepts.style2, PROC TEMPLATE uses that style element. (If there were no such element in concepts.style2, then PROC TEMPLATE would look for one in concepts.style1 and use that one.) Therefore, celldatalarge inherits the new definition of celldataemphasis, which includes the white background. Similarly celldatasmall, which now inherits from celldatalarge in concepts.style2, inherits the white background.

To make a change in a child style definition that is passed in turn to the style elements that are defined in the parent, and that inherit from the style element that you redefine in the child style definition, then use the REPLACE statement. You can only use the REPLACE statement if you have specified a parent style definition. The following program changes the background color of celldataemphasis by using a REPLACE statement. You can think of this REPLACE statement as replacing the statement that defines the like-named style element in concepts.style1. The REPLACE statement does not actually change the concepts.style1, but PROC TEMPLATE builds concepts.style2 as if it had changed concepts.style1.
Some code and explanation follows:

```plaintext
font_weight=bold
font_size=5;
end;
run;

proc template:
define style concepts.style2;
parent=concepts.style1;
    replace celldataemphasis from celldatasimple /
        style celldatasmall from celldatalarge /
        font_size=2;
end;
run;
```

This is how PROC TEMPLATE constructs `concepts.style2`:

1. The `PARENT=` attribute makes `concepts.style1` the basis of the new style definition. `concepts.style2` contains all the style elements that `concepts.style1` contains: `celldatasimple`, `celldataemphasis`, and `celldatalarge`.

2. `concepts.style2` does nothing to `celldatasimple`. Therefore, in `concepts.style2`, `celldatasimple` is the same as it is in `concepts.style1`.

3. The `REPLACE` statement essentially replaces the definition of `celldataemphasis` in `concepts.style1` while `concepts.style2` is being created. (It does not really alter `concepts.style1`, but `concepts.style2` is created as if it had.) Thus, not only does `celldataemphasis` now exist in `concepts.style2`, but also every style element that `concepts.style2` inherits from `concepts.style1` is based on the replaced definition.

A description of each style element in `concepts.style2` follows:

**celldatasimple**

is not redefined in `concepts.style2`. Nor does it inherit from any other style element. Therefore, it has the same attributes as `celldatasimple` in `concepts.style1`:

- FONT_FACE=arial
- BACKGROUND=very light vivid blue
- FOREGROUND=white.

**celldataemphasis**

is defined in `concepts.style2`. It inherits from `celldatasimple`, so initially it has these attributes:

- FONT_FACE=arial
- BACKGROUND=very light vivid blue
- FOREGROUND=white.

However, the `REPLACE` statement that creates `celldataemphasis` specifies a foreground color, a background color, and a font style. The foreground and background color specifications override the inherited attributes. Therefore, the final list of attributes for `celldataemphasis` is:

- FONT_FACE=arial
- BACKGROUND=white
- FOREGROUND=blue
- FONT_STYLE=italic.
*celldatalarge*

is not redefined in *concepts.style2*. Therefore, *concepts.style2* uses the same definition as *concepts.style1* uses. The definition of *celldatalarge* is from *celldataemphasis*. Because *celldataemphasis* was created in *concepts.style2* with a REPLACE statement, *celldatalarge* inherits the following attributes from the replaced definition of *celldataemphasis*:

- FONT_FACE=arial (from *concepts.style1*)
- BACKGROUND=white (from *concepts.style2*)
- FOREGROUND=blue (from *concepts.style2*)
- FONT_STYLE=italic (from *concepts.style2*).

The definition of *celldatalarge* from *concepts.style1* adds these attributes:

- FONT_WEIGHT=bold (from *concepts.style1*)
- FONT_SIZE=5 (from *concepts.style1*).

*celldatasmall*

exists only in *concepts.style2*. It is created from *celldatalarge*. PROC TEMPLATE first looks for *celldatalarge* in *concepts.style2*, but because it does not exist, it uses the definition in the parent style definition. Therefore, *celldatasmall* is just like *celldatalarge* except that the font size of 2 replaces the font size of 5. The final list of attributes for *celldatasmall* is

- FONT_FACE=arial
  BACKGROUND=white
  FOREGROUND=blue
  FONT_STYLE=italic
  FONT_SIZE=2

The following HTML output uses this new style definition, *concepts.style2*:
Display 9.11 Using the REPLACE Statement to Alter a Style Element and Its Children

<table>
<thead>
<tr>
<th>Country</th>
<th>Grain</th>
<th>Kilotons</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Rice</td>
<td>10035</td>
<td>10035</td>
</tr>
<tr>
<td>China</td>
<td>Rice</td>
<td>190100</td>
<td>190100</td>
</tr>
<tr>
<td>India</td>
<td>Rice</td>
<td>120012</td>
<td>120012</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Rice</td>
<td>51165</td>
<td>51165</td>
</tr>
<tr>
<td>United States</td>
<td>Rice</td>
<td>7771</td>
<td>7771</td>
</tr>
</tbody>
</table>

Summary of Style Element Inheritance in a Style Definition with a Parent

The following points summarize style element inheritance in a style definition that has a parent:

- You can create a new style element from any style element in the parent or the child style definition.
- If you create a style element from another style element, then PROC TEMPLATE first looks in the current style definition for that element. If the style definition does not contain such an element, then PROC TEMPLATE looks in the parent (and in parent’s parent, and so on).
- A new style element inherits all the attributes from its parent.
- You can specify additional attributes in the new style definition. The attributes are added to the attributes that the element inherits.
- You can change the value of an inherited attribute by respecifying it in the definition of the new style element.
- If you want to create a style element in the new style definition, then you must use the STYLE statement or the REPLACE statement.
  - In the STYLE statement, you are creating a new style element. Only those style elements that explicitly inherit their attributes from the style element that you created inherit the changes. All other style elements inherit their attributes from the parent style definition.
  - In the REPLACE statement, you are replacing the like–named style element from the parent style definition in the new style definition. The REPLACE statement does not change the parent style definition. All style elements that inherit attributes from the style elements that inherit style elements that you created, inherit only the attributes that you specified in the REPLACE statement. All other attributes specified in the parent style definition are not used unless you specify them again in the style element that you created.
Examples: Creating and Modifying Styles Using the TEMPLATE Procedure

Example 1: Creating a Stand-Alone Style Definition

PROC TEMPLATE features:
  DEFINE STYLE statement
    STYLE statement
      BACKGROUND=
      BORDERWIDTH=
      CELLSPACING=
      FONT_FACE=
      FONT_SIZE=
      FONT_STYLE=
      FONT_WEIGHT=
      FOREGROUND=
  DEFINE TABLE statement
    CLASSLEVELS= table attribute
    DYNAMIC statement
    MVAR statement
  DEFINE COLUMN statement
    BLANK_DUPS=
    GENERIC=
    HEADER=
    STYLE=
  DEFINE FOOTER statement
    TEXT statement

Other ODS features:
  ODS HTML statement
  ODS LISTING statement
  FILE statement with ODS= option
  PUT statement with _ODS_ argument

Data set:  GRAIN_PRODUCTION on page 97
Format:  $CNTRY. on page 98

Program Description

This example creates a style definition that is not based on any other style definition. When you create a style definition, you will usually base it on one of the definitions that SAS provides (see Example 3 on page 355). However, this example is provided to show you some of the basic ways to create a style definition.

It is important to understand that by default, certain table elements are created with certain style elements. For example, unless you specify a different style element with
the STYLE= attribute, ODS produces SAS titles with the `systemtitle` style element. Similarly, unless you specify otherwise, ODS produces headers with the `header` style element. (For information about each style element, see Appendix 4, “HTML, Printer Family, and Markup Languages Style Elements and Their Inheritances,” on page 651.

**Program**

Create a new style definition `newstyle` with the style element `cellcontents`. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style definition called `newstyle`. This STYLE statement defines the style element `cellcontents`. This style element is composed of the style attributes that appear on the STYLE statement. The FONT_FACE= 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 /
    background=blue
    foreground=white
    font_face="arial, helvetica"
    font_weight=medium
    font_style=roman
    font_size=4;
```

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 headers. This style element uses different foreground and background colors from `cellcontents`. It uses the same font (Arial or Helvetica) and the same font style (roman) as `cellcontents`. However, it uses a bold font weight and a large font size.

```
style header /
  background=very light blue
  foreground=blue
  font_face="arial, helvetica"
  font_weight=bold
  font_style=roman
  font_size=5;
```

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`, but it adds an italic font style and uses a larger font size.

```
style systemtitle /
  background=white
  foreground=red
  font_face="arial, helvetica"
  font_weight=bold
  font_style=italic
  font_size=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 FONT_SIZE= attribute in its definition.
Create the style element `table`. This STYLE statement creates the style element `table`. By default, ODS uses this style element to display tables.

```plaintext
style table /  
  cellspacing=5  
  borderwidth=10;

End the style definition. The END statement ends the style definition. The RUN statement executes the TEMPLATE procedure.

```plaintext
end;
run;
```

Create the table definition `table1`. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE TABLE statement creates a new table definition called `table1`.

```plaintext
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 definition to the data component to produce an output object. `SYSDATE9` is an automatic macro variable whose value is always available.

```plaintext
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 definition 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 headers gives you more flexibility than does hard-coding the headers in the table definition.

```plaintext
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, you should set this attribute as well.

```plaintext
classlevels=on;
```
Create the column char_var. This DEFINE statement and its attributes create the column definition `char_var`.

`GENERIC=` specifies that multiple variables can use the same column definition.

`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 definition is `cellcontents`.

The END statement ends the definition.

```plaintext
define column char_var;
  generic=on;
  blank_dups=on;
  header=colhd;
  style=cellcontents;
end;
```

Create the column definition num_var. This DEFINE statement and its attributes create the column definition `num_var`. `GENERIC=` specifies that multiple variables can use the same column definition. `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 definition is `cellcontents`.

The END statement ends the definition.

```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 definition (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 definition. This END statement ends the table definition. The RUN statement executes the PROC TEMPLATE step.

```plaintext
define column char_var;
  generic=on;
  blank_dups=on;
  header=colhd;
  style=cellcontents;
end;
```

Stop the creation of the listing output. The ODS LISTING statement closes the Listing destination in order to conserve resources. The Listing destination is open by default.

```plaintext
ods listing close;
```
Create HTML output and specify the location for storing the HTML output. Specify the style definition that you want to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file newstyle-body.htm in the current directory. The STYLE= option tells ODS to use newstyle as the style definition when it formats the output.

```ods html body=`newstyle-body.htm` style=newstyle;```

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

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

Create the data component. This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object.

The SET statement reads the data set GRAIN_PRODUCTION. The WHERE statement subsets the data set so that the output object contains information only for rice and corn production in 1996.

```data _null_;
set grain_production;
where type in ('Rice', 'Corn') and year=1996;
```

Route the DATA step results to ODS and use the table1 table definition. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, “Output Delivery System and the DATA Step,” on page 39.) The TEMPLATE= suboption tells ODS to use the table definition named table1, which was previously created with PROC TEMPLATE.

```file print ods=(
    template=`table1`
```

Specify the column definition to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table definition. 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 definition named char_var. GENERIC= must be set to ON in both the table definition and each column assignment in order for multiple variables to use the same column definition. 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 definition, 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.

```plaintext
   put _ods_;
run;
```

Stop the creation of the HTML output and create the listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser. The ODS LISTING statement opens the Listing destination to return ODS to its default setup.

```plaintext
ods html close;
ods listing;
```

**HTML Output: Specifying Colors and Fonts with User-Defined Attributes**

**Display 9.12**  HTML Output (Viewed with Microsoft Internet Explorer)

You can use the fonts to confirm that SAS titles use the `systemtitle` style element, that column headers use the `header` style element, that the footer uses the `table-footer` style element, and that the contents of both character and numeric cells use the `cellcontents` style element. Use the width of the table border and the spacing between cells to confirm that the table itself is produced with the `table` style element.

---

**Leading Grain Producers in 1996**

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil</td>
<td>Corn</td>
<td>31,975</td>
</tr>
<tr>
<td></td>
<td>Rice</td>
<td>10,035</td>
</tr>
<tr>
<td>China</td>
<td>Corn</td>
<td>119,350</td>
</tr>
<tr>
<td></td>
<td>Rice</td>
<td>190,100</td>
</tr>
<tr>
<td>India</td>
<td>Corn</td>
<td>8,660</td>
</tr>
<tr>
<td></td>
<td>Rice</td>
<td>120,012</td>
</tr>
<tr>
<td>Indonesia</td>
<td>Corn</td>
<td>8,926</td>
</tr>
<tr>
<td></td>
<td>Rice</td>
<td>51,165</td>
</tr>
<tr>
<td>United States</td>
<td>Corn</td>
<td>236,064</td>
</tr>
<tr>
<td></td>
<td>Rice</td>
<td>7,771</td>
</tr>
</tbody>
</table>

*Prepared on 07APR2003*
Example 2: Creating and Modifying a Style Definition with User-Defined Attributes

PROC TEMPLATE features:
- DEFINE STYLE statement
  - STYLE statement with user-defined attributes
- DEFINE TABLE statement
  - CLASSLEVELS= table attribute
- DYNAMIC statement
- MVAR statement
- DEFINE COLUMN statement
  - BLANK_DUPS=
  - GENERIC=
  - HEADER=
  - STYLE=
- DEFINE FOOTER statement
- TEXT statement

Other ODS features:
- ODS HTML statement
- ODS LISTING statement
- FILE statement with ODS= option
- PUT statement with _ODS_ argument

Data set: GRAIN_PRODUCTION“Program” on page 97.
Format: $CNTRY. on page 98

Program 1: Description

This example creates a style definition that is equivalent to the style definition that Example 1 on page 342 creates. However, this style definition uses user-defined attributes to specify colors and fonts. This technique makes it possible to easily make changes in multiple places in your output.

Program 1: Creating the Style Definition

Create the style definition newstyle2. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style definition called newstyle2. This STYLE statement defines the style element fonts.

This style element is composed of three user-defined attributes: cellfont, headingfont, and titlefont. Each of these attributes describes a font. This style definition specifies the font_face, font_size, font_weight, and the font_style for each of the three attributes. The font and font_width attributes are still defined by the default style definition.
proc template;
  define style newstyle2;
    style fonts /
      "cellfont"=("arial, helvetica", 4, medium roman)
      "headingfont"=("arial, helvetica", 5, bold roman)
      "titlefont"=("arial, helvetica", 6, bold italic);

Create the style element colors. This STYLE statement defines the style element colors. This style element is composed of four user-defined attributes: light, medium, dark, and bright. The values for medium and dark are RGB values equivalent to very light blue and blue.

    style colors /
      "light"=white
      "medium"=cxaaff
      "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 definition. For example, the foreground color in cellcontents is set to colors("light"). Looking at the definition 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 /
      background=colors("dark")
      foreground=colors("light")
      font=fonts("cellfont");
    style header /
      background=colors("medium")
      foreground=colors("dark")
      font=fonts("headingfont");
    style systemtitle /
      background=colors("light")
      foreground=colors("bright")
      font=fonts("titlefont");
    style footer from systemtitle /
      font_size=3;
    style table /
      cellspacing=5
      borderwidth=10;

End the style definition. The END statement ends the style definition. The RUN statement executes PROC TEMPLATE.

    end;
    run;

Create the table definition table1. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE TABLE statement creates a new table definition 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 definition to the data component to produce an output object. `SYSDATE9` is an automatic macro variable whose value is always available.

```plaintext
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 definition 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 headers gives you more flexibility than hard-coding the headers in the table definition does.

```plaintext
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, you should set this attribute as well.

```plaintext
classlevels=on;
```

Create the column `char_var`. This `DEFINE` statement and its attributes create the column definition `char_var`.

- `GENERIC=` specifies that multiple variables can use the same column definition.
- `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 definition is `cellcontents`.

The END statement ends the definition.

```plaintext
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 definition `num_var`.

- `GENERIC=` specifies that multiple variables can use the same column definition.
- `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 definition is `cellcontents`.

The END statement ends the definition.

```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 definition (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 definition. This END statement ends the table definition. The RUN statement executes the PROC TEMPLATE step.

```plaintext
end;
run;
```

Stop the creation of the listing output. The ODS LISTING statement closes the Listing destination to conserve resources. The Listing destination is open by default.

```plaintext
ods listing close;
```

Create the HTML output and specify the style definition that you want to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file **newstyle2-body.htm** in the current directory. The STYLE= option tells ODS to use **newstyle2** as the style definition when it formats the output.

```plaintext
ods html body='newstyle2-body.htm'
  style=newstyle2;
```

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 definition. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, “Output Delivery System and the DATA Step,” on page 39. The TEMPLATE= suboption tells ODS to use the table definition named **table1**, which was previously created with PROC TEMPLATE.

```plaintext
file print ods=(
  template='table1'
```
Specify the column definition to use for each variable. The COLUMNS= suboption places
DATA step variables into columns that are defined in the table definition. 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 definition named char_var. GENERIC=
must be set to ON in both the table definition and each column assignment in order for multiple
variables to use the same column definition. 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 definition, the column header is **Year**.

```plaintext
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.

```plaintext
put _ods_;
run;
```

Stop the creation of the HTML output and create the listing output. The ODS HTML
statement closes the HTML destination and all the files that are associated with it. You must
close the destination before you can view the output with a browser. The ODS LISTING
statement opens the Listing destination to return ODS to its default setup.

```plaintext
ods html close;
ods listing;
```
Original HTML Output

Display 9.13  HTML Output (Viewed with Microsoft Internet Explorer)

This HTML output is identical to “HTML Output: Specifying Colors and Fonts with User-Defined Attributes” on page 347, which was produced with a style definition that used predefined style attributes. You can use the fonts to confirm that SAS titles use the `systemtitle` style element, that column headers 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.

Program 2: Description

In the program Example 1 on page 342, if you want to change the color scheme so that the blues are replaced by pink and red, then you must change each occurrence of “blue” and “very light blue.” In this program, because colors are defined as user-defined attributes, you need to make the change only once.

Program 2: Changing User-Defined Attributes

To make the color scheme change, you need to change only the following section of code:

```plaintext
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`, you can change the following section of code:

```
"cellfont"=("arial, helvetica", 4, medium roman)
```

Here is one example of how you can change the code:

```
"cellfont"=("courier, arial, helvetica", 4, medium roman)
```

The following HTML output shows the results of running the same program with these changes.

**HTML Output: Changing Colors and Fonts of User-Defined Attributes**

**Display 9.14**  HTML Output with Changed Colors and Fonts (Viewed with Microsoft Internet Explorer)

You can see that the font that is used 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 Definition for the HTML and Markup Languages

PROC TEMPLATE features:
- DEFINE STYLE statement
- PARENT= attribute
- REPLACE statement
- style attributes
  - user-defined attributes
  - BACKGROUND=
  - BORDERWIDTH=
  - CELLPADDING=
  - CELLSPIXING=
  - FONT=
  - FONT_STYLE=
  - FOREGROUND=
  - FRAME=
  - POSTHTML=
  - RULES=

Other ODS features:
- ODS HTML statement
  - STYLE= option
- ODS LISTING statement
- ODS PATH statement

Data set: ENERGY “Program” on page 138
Formats: DIVFMT. and USETYPE. on page 139

Program 1: Description

When you are working with style definitions, you are more likely to modify a SAS style definition than to write a completely new style definition. This example shows you how to make changes to the default style definition for the HTML destination. The new style definition affects both the contents file and the body file in the HTML output. In the contents file, the modified style definition 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 definition 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 headers. 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 such as cellspacing, rules, and border width.
Note: Remember that when a STYLE statement creates a style element in the new style definition, only style elements that explicitly inherit from that style element in the new definition will inherit the change. When a REPLACE statement creates a style element in the new style definition, all style elements that inherit from that element inherit the definition that is in the new style definition, so the change appears in all children of the element.

Program 1: Using the Default Style Definition with PROC PRINT

Specify the search path in order to locate the table definition. This statement specifies which locations to search for definitions that were created by PROC TEMPLATE, as well as the order in which to search for them. The statement is included to ensure that the example works correctly. However, if you have not changed the path, then you do not need to include this statement because it specifies the default path.

ods path sasuser.templat(update) sashelp.tmplmst(read);

Stop the creation of the listing output. The ODS LISTING statement closes the Listing destination to conserve resources. The Listing destination is open by default.

ods listing close;

Create the HTML output and specify the name of the HTML file. Specify the style definition that you want to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. The output from PROC PRINT is sent to the body file. FRAME= and CONTENTS= create a frame that includes a table of contents that links to the contents of the body file. The body file also appears in the frame.

The STYLE= option tells ODS to use styles.default as the style definition when it formats the output. Strictly speaking, this option is unnecessary because it specifies the default style definition, but it is included for clarity.

ods html body='sasdefaultstyle-body.htm'
   contents='sasdefaultstyle-content.htm'
   frame='sasdefaultstyle-frame.htm'
   style=styles.default;

Specify the titles and footnote for the report. The TITLE and FOOTNOTE statements provide two titles and a footnote for the output. The FOOTNOTE statement uses double rather than single quotes so that the macro variable resolves.

title 'Energy Expenditures for Each Region';
title2 ' (millions of dollars)';
footnote "Report prepared on &sysdate9";

Print the report. PROC PRINT creates a report that includes three variables. ODS writes the report to the BODY file.

proc print data=energy noobs;
   var state type expenditures;
   format division divfmt. type usetype. expenditures comma12.;
   by division;
   where division=2 or division=3;
run;
Program 2: Modifying the Default Style Definition and Using It with PROC PRINT

Create the style definition customdefault. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style definition called customdefault.

```latex
proc template;
   define style customdefault;
```

Specify the parent style definition from which the customdefault style definition inherits its attributes. The PARENT= attribute specifies styles.default as the style definition from which the current style definition inherits. All the style elements, attributes, and statements that are specified in the parent’s definition are used in the current definition unless the current definition overrides them.

```latex
parent=styles.default;
```

Change the attributes of the style element color_list. This REPLACE statement adds to the child style definition the style element color_list, which also exists in the parent style definition. You can think of the REPLACE statement as replacing the definition of color_list in the parent style definition. The REPLACE statement does not actually change the parent style definition, but PROC TEMPLATE builds the child style definition as if it had changed the parent. All style elements that use the user-defined attributes that color_list defines (fgB2, fgB1, etc.) use the attributes that are specified in the REPLACE statement, not the ones that are specified in styles.default. Therefore, if you change a color here, then you change every occurrence of the color in the HTML output. This REPLACE statement changes the values of fgA2 and fgA from a greenish blue to a pure blue and from a slightly darker greenish blue to a purple. (The first two digits of the hex value represent red, the next two represent green, and the last two represent blue.)

```latex
replace color_list /
   'fgB2' = cx0066AA
```
Change the attributes of the style element `titlesandfooters`. This REPLACE statement adds to the child style definition the style element `titlesandfooters`, which also exists in the parent style definition. The new definition does not inherit attributes from any style element, but it will pass its attributes to any style element that inherits from `titlesandfooters` or from a child of `titlesandfooters`. This style element uses `systitlefg` and `systitlebg` for colors, but it changes the font size from the relative size of 4 that is specified in `titlefont2` to a relative size of 3. As a result, the titles and footnotes in Display 9.16 on page 361 are smaller than the ones in Display 9.15 on page 357.

```latex
replace titlesandfooters /
  foreground=colors("systitlefg")
  background=colors("systitlebg")
  font=fonts("titlefont2") font_size=3;
```

Change the attributes of the style element `blyline`. Specify that the style element `blyline` inherits its attributes from the `titlesandfooters` style element. This REPLACE statement adds to the child style definition the style element `blyline`, which also exists in the parent style definition. This style element inherits all attributes from `titlesandfooters` as it is specified in the previous REPLACE statement. Therefore, the initial definition for the `blyline` includes the foreground and background colors that are used for system titles, and a smaller version of `titlefont2`. However, the FOREGROUND= attribute replaces the foreground color with the foreground color that is used for headers. Note that in the default style definition, the background color for the `blyline` differs from the background color for the document, so it appears as a gray stripe in Display 9.15 on page 357. In this customized style definition, the stripe disappears because the background color for the `blyline` and the document are the same.

```latex
replace blyline from titlesandfooters /
  foreground=colors("headerfg");
```

Change one attribute in the style definition `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 definition. The change does not affect `headerfixed` and the other style elements that inherit from `header` in the parent style definition.

```latex
style header from header /
  font_style=italic;
```
Customize the text that is used in parts of the output. This REPLACE statement alters the text that is used in parts of the HTML output. In the contents file, the default style definition uses “The” as the value of `prefix1` and “Procedure” as the value of `suffix1`. Thus, in HTML output that uses the default style definition, the output from PROC PRINT is identified by “1. The PRINT Procedure” (see Display 9.15 on page 357). In the customized style definition, 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 definition in a variety of circumstances.)

```
replace text /
  "prefix1" = "PROC 
  "suffix1" = ":"
  "Content Title" = "Contents"
  "Pages Title" = "Pages"
  "Note Banner" = "Note:"
  "Warn Banner" = "Warning:
  "Error Banner" = "Error:"
  "Fatal Banner" = "Fatal:"
;
```

Customize the presentation of the HTML table that contains the output from PROC PRINT. This STYLE statement changes the presentation of the HTML table that contains the output from PROC PRINT. The background color, the kind of box that surrounds the table, and the cell padding remain the same as in `styles.default`, but all the other attributes are changed. RULES=COLS draws rules only between the columns of the table. CELLSPACING=0 removes the spacing between the cells of the table so that the data appear on a continuous background. BORDERWIDTH= increases the width of the table’s border. The changes dramatically alter the appearance of the HTML output.

```
style table from table /
  rules=cols
  cellspacing=0
  borderwidth=5;
```

Change the color of links and the foreground. This STYLE statement changes the value of the VISIBLELINKCOLOR= attribute in the style element `contents` so that the links in the table of contents appear in the same color as the rest of the table of contents. It also changes the foreground color so that the title of the table of contents appears in the same color as system titles.

```
style contents from contents /
  visitedlinkcolor=colors("systitlefg")
  foreground = colors('systitlefg');
```

Add more space between the items in the table of contents. This STYLE statement adds the POSTHTML= attribute so that the items in the table of contents are displayed with extra space between them.

```
style contentitem from contentitem /
  posthtml=’<p>’;
```
Stop the creation of the customized style definition. The END statement ends the style definition. The RUN statement executes the PROC TEMPLATE step.

```plaintext
end;
run;
```

Create the HTML output and specify the style definition that you want 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 definition when it formats the output.

```plaintext
ods html body='customdefaultstyle-body.htm'
   contents='customdefaultstyle-content.htm'
   frame='customdefaultstyle-frame.htm'
   style=customdefault;
```

Specify the titles and footnote for the report. The TITLE and FOOTNOTE statements provide two titles and a footnote for the output. The FOOTNOTE statement uses double rather than single quotes so that the macro variable resolves.

```plaintext
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
footnote "Report prepared on &sysdate9";
```

Print the customized report. PROC PRINT creates a report that includes three variables. ODS writes the report to the body file. This PROC PRINT step is the same one that was used with the default style definition earlier.

```plaintext
proc print data=energy noobs;
   var state type expenditures;
   format division divfmt. type usetype. expenditures comma12.;
   by division;
   where division=2 or division=3;
run;
```

Stop the creation of the HTML output and initiate the creation of listing output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser. The ODS LISTING statement opens the Listing destination to return ODS to its default setup.

```plaintext
ods html close;
ods listing;
```
Example 4: Defining a Table and Graph Style

PROC TEMPLATE features:
DEFINE STYLE statement
    PARENT= attribute
    REPLACE statement
style attributes

Display 9.16  HTML Output from PROC PRINT with the Customized Style Definition

Energy Expenditures for Each Region
(millions of dollars)

**Division—Middle Atlantic**

<table>
<thead>
<tr>
<th>State</th>
<th>Type</th>
<th>Expenditures</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>Residential Customers</td>
<td>8,765</td>
</tr>
<tr>
<td>NY</td>
<td>Business Customers</td>
<td>7,825</td>
</tr>
<tr>
<td>NJ</td>
<td>Residential Customers</td>
<td>4,115</td>
</tr>
<tr>
<td>NJ</td>
<td>Business Customers</td>
<td>3,539</td>
</tr>
<tr>
<td>PA</td>
<td>Residential Customers</td>
<td>6,478</td>
</tr>
<tr>
<td>PA</td>
<td>Business Customers</td>
<td>3,695</td>
</tr>
</tbody>
</table>

**Division—Mountain**

<table>
<thead>
<tr>
<th>State</th>
<th>Type</th>
<th>Expenditures</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>Residential Customers</td>
<td>322</td>
</tr>
<tr>
<td>MT</td>
<td>Business Customers</td>
<td>232</td>
</tr>
<tr>
<td>ID</td>
<td>Residential Customers</td>
<td>392</td>
</tr>
<tr>
<td>ID</td>
<td>Business Customers</td>
<td>298</td>
</tr>
<tr>
<td>WY</td>
<td>Residential Customers</td>
<td>164</td>
</tr>
<tr>
<td>WY</td>
<td>Business Customers</td>
<td>164</td>
</tr>
<tr>
<td>CO</td>
<td>Residential Customers</td>
<td>1,215</td>
</tr>
<tr>
<td>CO</td>
<td>Business Customers</td>
<td>1,173</td>
</tr>
<tr>
<td>NM</td>
<td>Residential Customers</td>
<td>645</td>
</tr>
<tr>
<td>NM</td>
<td>Business Customers</td>
<td>678</td>
</tr>
<tr>
<td>AZ</td>
<td>Residential Customers</td>
<td>1,694</td>
</tr>
<tr>
<td>AZ</td>
<td>Business Customers</td>
<td>1,440</td>
</tr>
<tr>
<td>UT</td>
<td>Residential Customers</td>
<td>621</td>
</tr>
<tr>
<td>UT</td>
<td>Business Customers</td>
<td>439</td>
</tr>
<tr>
<td>NV</td>
<td>Residential Customers</td>
<td>883</td>
</tr>
<tr>
<td>NV</td>
<td>Business Customers</td>
<td>378</td>
</tr>
</tbody>
</table>
When you are working with style definitions, you are more likely to modify a SAS style definition than to write a completely new style definition. 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 definition, only style elements that explicitly inherit from that style element in the new definition inherit the change. When a REPLACE statement creates a style element in the new style definition, all style elements that inherit from that element inherit the definition that is in the new style definition, so the change appears in all children of the element.

Program

Create the style definition Science. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style definition in the STYLES catalog called Science.

```
proc template;
  define style Styles.Science;
```
Specify the parent style definition from where the SCIENCE style definitions inherits its attributes. The PARENT= attribute specifies STYLES.DEFAULT as the style definition that the current style definition inherits from. All the style elements and attributes and statements that are specified in the parent’s definition are used in the current definition unless the current definition overrides them.

```plaintext
parent = styles.default;
```

Change the attributes fonts in the parent style definition by replacing it in the child style definition Science. The REPLACE statement adds to the child style definitions the style elements fonts, which also exist in the parent style definitions. While the REPLACE statement does not actually change the parent definition, PROC TEMPLATE builds the child style definition as if it had changed the parent. All style elements that use the user-defined attributes that fonts define use the attributes that are specified in the REPLACE statements, not the ones that are specified in the STYLES.DEFAULT style definition.

```plaintext
replace 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)
   '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 REPLACE statement adds to the child style definitions the style elements GraphFonts, which also exist in the parent style definitions. While the REPLACE statement does not actually change the parent definition, PROC TEMPLATE builds the child style definition as if it had changed the parent. All the style elements that use the user-defined attributes that GraphFonts define use the attributes specified in the REPLACE statement, not those specified in STYLES.DEFAULT style definition.

```plaintext
replace GraphFonts /
   'GraphValueFont' = ("Verdana",10pt)
   'GraphLabelFont' = ("Verdana",14pt,Bold);
```

Change the attributes colors in the parent style definition by replacing it in the child style definition Science. The REPLACE statement adds to the child style definitions the style elements colors, which also exist in the parent style definitions. While the REPLACE statement does not actually change the parent definition, PROC TEMPLATE builds the child style definition as if it had changed the parent. All style elements that use the user-defined attributes that colors define use the attributes that are specified in the REPLACE statements, not the ones that are specified in STYLES.DEFAULT style definition.

```plaintext
replace colors /
   'headerfgemph' = cx31035E
   'headerbgemph' = cxFFFFFF
```
Change the attributes for graph style specific colors. The REPLACE statement adds to the child style definitions the style elements GraphColors, which also exist in the parent style definitions. While the REPLACE statement does not actually change the parent definition, PROC TEMPLATE builds the child style definition as if it had changed the parent. All the style elements that use the user-defined attributes that GraphColors define use the attributes that are specified in the REPLACE statement, not the attributes that are specified in STYLES.DEFAULT.

```
replace GraphColors /
  'gconramp3cend' = cxDD6060
  'gconramp3cneutral' = cFFFFFFF
  'gconramp3cstart' = cx6497EB
  'gramp3cend' = cxBED8D3
  'gramp3cneutral' = cFFFFFFF
  'gramp3cstart' = cxAAB6DF
  'gconramp2cend' = cx6497EB
  'gconramp2cstart' = cFFFFFFF
```
'gramp2cend' = cX548287
'gramp2cstart' = cXFFFFFF
'gtext' = cX31035E
'glabel' = cX31035E
'gborderlines' = cX31035E
'goutlines' = cX31035E
'ggrid' = cX31035E
'gaxis' = cX31035E
'gshadow' = cX707671
'glegend' = cXFFFFFF
'gfloors' = cXDFFEE
'gwalls' = cXFFFFFF
'gcdata12' = cXFF667F
'gcdata11' = cX5050CC
'gcdata10' = cXE100BF
'gcdata9' = cX007F00
'gcdata8' = cXB99600
'gcdata7' = cX7F7F7F
'gcdata6' = cX984EA3
'gcdata5' = cX4DABF4A
'gcdata4' = cxA65628
'gcdata3' = cXFF7F00
'gcdata2' = cX377DB8
'gcdata1' = cXE31A1C
'gcdata12' = cX8A5573
'gcdata11' = cXCFB1E2
'gcdata10' = cX88E829D
'gcdata9' = cX2952B1
'gcdata8' = cXAA66DF
'gcdata7' = cX6771C2
'gcdata6' = cXBED8D3
'gcdata5' = cX8B65A3
'gcdata4' = cXB0D3AB
'gcdata3' = cX4B8287
'gcdata2' = cX7DC1C9
'gcdata1' = cX9580D5;

Specify attributes for the table. This STYLE statement is applied to tables. This statement specifies a cell padding of 5 and a cell spacing of 2, that the BORDERCOLOR, TABLEBORDERCOLOR, which is cX909090, and that the BORDERCOLORLIGHT, TABLEBORDERLIGHT, which is cXFFFFFF, should blend to create the table border color, and sets a BORDERWIDTH of 2. 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 style definitions.

    style Table from Output /
        cellpadding = 5
        cellspacing = 2
        bordercolordark = colors('tableborderdark')
        bordercolorlight = colors('tableborderlight')
        bordervdith = 2;

Specify attributes for the GraphLabelText element. This STYLE statement is applied to the graph's label text. A DROPSHADOW attribute is applied.
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
replace GraphBackground
"Graph background attributes" /
background = colors('docbg')
image = "//dntsrs/sas/m900/ods/misc/Science.gif"
just = L
vjust = B;
```

Specify attributes for the GraphAxisLines element. This STYLE statement is applied to the graph’s axis line. The **OUTPUTWIDTH** is 2.

```plaintext
style GraphAxisLines from GraphAxisLines
"Axis line attributes" /
outputwidth = 2;
```

Specify attributes for the GraphBorderLines element. This STYLE statement is applied to the borderlines in the graph. The **OUTPUTWIDTH** is 2 and the **FOREGROUND** color defined in **gaxis**, which is CX31035E, is used.

```plaintext
style GraphBorderLines from GraphBorderLines
"Border attributes" /
outputwidth = 2
foreground = colors('gaxis');
```

Specify attributes for the GraphCharts element. This STYLE statement is applied to the graph’s chart. The data elements of the graph have a **TRANSPARENCY** of 25 percent.

```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 of the graph’s axes. The **GRADIENT_DIRECTION** is set to **Xaxis**, meaning the gradient is going left to right. The **ENDCOLOR**, defined in **gwalls**, which is CXFFFFFF, is the final color used with the gradient. The data elements of the graph have a **TRANSPARENCY** of 100 percent. Since a **STARTCOLOR** is not specified, the beginning of the gradient is completely transparent.

```plaintext
style GraphWalls from GraphWalls
"Wall Attributes" /
gradient_direction = "Xaxis"
endcolor = colors('gwalls')
transparency = 1.0
```

Add the style to the specified catalog. The END statement ends the style definition. The **RUN** statement executes the PROC TEMPLATE step.

```plaintext
end;
run;
```
Overview: ODS Tabular Output

Why Use the TEMPLATE Procedure to Create or Customize Tabular Output?

The TEMPLATE procedure enables you to customize the tabular appearance of your SAS output. With the TEMPLATE procedure, you can create and modify table definitions, column definitions, header definitions, and footer definitions. The Output Delivery System then uses these definitions to produce customized tabular output for better data presentations and reports than what you get with the default SAS output.

By default, ODS output is formatted according to the various definitions that the procedure or DATA step specify. However, you can customize your tabular output definitions, or create your own new tabular output definitions, by using the TEMPLATE procedure with the following statements.
Terminology

For definitions of terms used in this section, see “Terminology: TEMPLATE Procedure” on page 266.

What Can You Do with a Table Definition?

Default Listing and RTF Display of an Output Object

By default, ODS uses the table definitions specified by the procedure or DATA step to create ODS output. For example, the following display shows the default listing output of the Basic Statistical Measures output object created by PROC UNIVARIATE. The second display shows the default RTF output of the same output object. You can use PROC TEMPLATE to customize the Basic Statistical Measures table definition.

Output 10.1  Listing Output from PROC UNIVARIATE (Default Basic Statistical Measures Tables)

<table>
<thead>
<tr>
<th>Location</th>
<th>Variability</th>
<th>Location</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean 3.877020</td>
<td>Std Deviation 5.16465</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median 2.423000</td>
<td>Variance 26.67364</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mode .</td>
<td>Range 28.66500</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interquartile Range 3.60000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean 1.040429</td>
<td>Std Deviation 0.66036</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median 0.961000</td>
<td>Variance 0.43608</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mode 0.608000</td>
<td>Range 2.75600</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interquartile Range 1.12700</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Display 10.1  RTF Output of City Population Statistics from PROC UNIVARIATE (Default Basic Statistical Measures Tables)

<table>
<thead>
<tr>
<th>Basic Statistical Measures</th>
<th>Location</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3.97702</td>
<td>5.9663</td>
</tr>
<tr>
<td>Median</td>
<td>2.42103</td>
<td>26.37364</td>
</tr>
<tr>
<td>Mode</td>
<td>Range</td>
<td>28.6639</td>
</tr>
<tr>
<td></td>
<td>Interquartile Range</td>
<td>3.0098</td>
</tr>
</tbody>
</table>

Display 10.2  RTF Output of Non City Population Statistics from PROC UNIVARIATE (Default Basic Statistical Measures Tables)

<table>
<thead>
<tr>
<th>Basic Statistical Measures</th>
<th>Location</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.04629</td>
<td>0.66928</td>
</tr>
<tr>
<td>Median</td>
<td>0.981039</td>
<td>0.45683</td>
</tr>
<tr>
<td>Mode</td>
<td>Range</td>
<td>2.75640</td>
</tr>
<tr>
<td></td>
<td>Interquartile Range</td>
<td>1.15790</td>
</tr>
</tbody>
</table>

Customized Version of the Listing and RTF Display of an Output Object

With the TEMPLATE procedure, you can change many of the table elements and obtain a customized format for the output objects. Here are some of customizations that you can do.

- Change the color and the font of the text of the first table header.
- Change the justification of the first table header.
- Change the setting of the table attributes UNDERLINE and OVERLINE.
- Change the line spacing between the rows.

Note: Not all table definition changes affect all destinations. For example, font changes are ignored in the LISTING destination.
The following displays show the results using a customized table definition that changes the first table header attributes, sets underlining and overlining in the table, and changes the amount of spacing between rows.

**Output 10.2**  Listing Output from PROC UNIVARIATE (Customized Basic Statistical Measures Tables)

```
The SAS System
The UNIVARIATE Procedure
Variable: CityPop_90  (1990 metropolitan pop in millions)
Basic Statistical Measures

<table>
<thead>
<tr>
<th>Location</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3.877020</td>
</tr>
<tr>
<td>Median</td>
<td>2.423000</td>
</tr>
<tr>
<td>Mode</td>
<td>.</td>
</tr>
</tbody>
</table>

| Std Deviation | 5.16465     |
| Variance      | 26.67364    |
| Range         | 28.66500    |
| Interquartile Range | 3.60000 |

The SAS System
The UNIVARIATE Procedure
Variable: NonCityPop_90  (1990 nonmetropolitan pop in million)
Basic Statistical Measures

<table>
<thead>
<tr>
<th>Location</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.040429</td>
</tr>
<tr>
<td>Median</td>
<td>0.961000</td>
</tr>
<tr>
<td>Mode</td>
<td>0.608000</td>
</tr>
</tbody>
</table>

| Std Deviation | 0.66036 |
| Variance      | 0.43608 |
| Range         | 2.75600 |
| Interquartile Range | 1.12700 |
```
Comparing the Edit of an Existing Table Definition with Creating a New Table Definition

If you want to change a table definition without completely redefining it, then you use an EDIT statement. When you use the EDIT statement, you keep all the definitions and attributes that already exist in the table definition, and only change the definitions or attributes specified in the EDIT statement. By default, the modified table definition...
is stored in SASUSER.TEMPLAT with the same name as the table definition that you specified in the EDIT statement.

If you want to create a new table definition, then you use the DEFINE TABLE statement. A table definition cannot be a parent to itself because creating a table through inheritance causes a corrupt template store, and then the definition must be deleted. When you create a new table definition, only the columns, headers, footers, and table attributes that you define exist in the new table definition.

Note: If you edit an existing table, or define a new table with the same name as an existing table, then the table definition will be stored in the SASUSER.TEMPLAT item store and this table definition will be used, by default, unless you specify that the SASHELP.TMPLMST path is searched first.

---

Tabular Syntax: TEMPLATE Procedure

PROC TEMPLATE;

   EDIT definition-path-1 < AS definition-path-2 > < / STORE=libref.template-store > ;
   statements-and-attributes
   END;

   DEFINE COLUMN column-path < / STORE=libref.template-store >;
   statements-and-attributes
   END;

   DEFINE FOOTER footer-path < / STORE=libref.template-store >
   statements-and-attributes
   END;

   DEFINE HEADER definition-name;
   statements-and-attributes
   END;

   DEFINE TABLE table-path < / STORE=libref.template-store >;
   statements-and-attributes
   END;

The following table lists the statements that you use to add different features to your SAS tabular output.

Table 10.1 PROC TEMPLATE Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit an existing definition for a table,</td>
<td>EDIT</td>
</tr>
<tr>
<td>column, header, or footer.</td>
<td></td>
</tr>
<tr>
<td>Create a column definition.</td>
<td>DEFINE COLUMN</td>
</tr>
<tr>
<td>Create a footer definition.</td>
<td>DEFINE FOOTER</td>
</tr>
<tr>
<td>Create a header definition.</td>
<td>DEFINE HEADER</td>
</tr>
<tr>
<td>Create a table definition.</td>
<td>DEFINE TABLE</td>
</tr>
</tbody>
</table>
EDIT Statement

Edits an existing definition for a table, column, header, or footer

**Requirement:** An END statement must follow the EDIT statement, after 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 definition, you can also edit one or more column, header, or footer definitions that are defined in the table.
- When you edit a column definition, you can also edit one or more header definitions that are defined for that column.

**Restriction:** If you edit a definition that is a link, you break the link and create a separate definition.

**Featured in:** Example 1 on page 515

```sas
EDIT definition-path-1 <AS definition-path-2 > </ STORE=libref.template-store>;
   attribute-statements;
END;
```

**Required Arguments**

**definition-path-1**
- specifies a definition to edit. `definition-path-1` 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.)

**Interaction:** The `STORE=` option specifies a particular template store to read from and to write to.

**Tip:** You can determine what definitions a procedure or DATA step uses by submitting the ODS TRACE ON statement before you run the SAS program (see “ODS TRACE Statement” on page 197).

**Options**

**AS definition-path-2**
- specifies the location in which to store the edited definition, where `definition-path-2` 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.) By default, PROC TEMPLATE writes the edited definition to the first template store that you can write to in the current path.

**Default:** If you do not specify `AS definition-path-2`, PROC TEMPLATE writes the edited definition to `definition-path-1` in the first template store that you can write to.

**Restriction:** If the current EDIT statement is inside a set of editing instructions, then you cannot use the `AS definition-path-2` option.
STORE=libref.template-store
specifies the template store from which to read definition-path-1 and in which to store definition-path-2.

Statements and Attributes

The EDIT statement supports the same statements and attributes as the DEFINE TABLE statements. For more information, see “DEFINE TABLE Statement” on page 410.

Editing an Existing Definition

There are two steps to follow when you edit an existing definition.

1. Open a copy of the specified file.
   By default, PROC TEMPLATE looks for definition-path-1 in the list of template stores that is defined by the PATH statement (see “PATH Statement” on page 276). It opens a copy of the first definition path that it finds in a template store that has read access.

2. Save the modified file.
   PROC TEMPLATE writes the modified definition to the first template store in the current path with update access. If you do not specify a second definition path to write to, then it uses definition-path-1. Therefore, if you have update access to the template store from which you read definition-path-1, then you actually modify the original definition. Otherwise, the modified file is written to a template store to which you do have update access.

   If you do specify a second definition path, then PROC TEMPLATE writes the edited definition to the specified path in the first template store to which you have write access.

---

DEFINE COLUMN Statement

Creates a definition for a column

Requirement: An END statement must be the last statement in the definition.

Interaction: A column definition can include one or more header definitions.

See also: “DEFINE HEADER Statement” on page 395

Featured in: Example 3 on page 528 and Example 5 on page 539

DEFINE COLUMN column-path< / STORE=libref.template-store>; <column-attribute-1; <… column-attribute-n; >>
   CELLSTYLE expression-1 AS <style-element-name><[style-attribute-specification(s)]>
   ><…, expression-n AS <style-element-name><[style-attribute-specification(s)]>>;
   COMPUTE AS expression;
   DEFINE HEADER definition-path;
   statements-and-attributes
   END;
   DYNAMIC variable-1<’text-1’> <… variable-n<’text-n’>>;
**MVAR** `variable-1<text-1'> <... variable-n<text-n'>>;  
**NMVAR** `variable-1<text-1'> <... variable-n<text-n'>>;  
**NOTES** `text';  
**TRANSLATE** `expression-1` **INTO** `expression-2` <... `expression-n` **INTO** `expression-m'>;  
**END**;

---

**Table 10.2** DEFINE COLUMN Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set one or more column attributes.</td>
<td><code>column-attributes</code></td>
</tr>
<tr>
<td>Set the style element of the cells in the column according to the values of the variables.</td>
<td>CELLSTYLE-AS</td>
</tr>
<tr>
<td>Compute values for a column that is not in the data component, or modify the values of a column that is in the data component.</td>
<td>COMPUTE AS</td>
</tr>
<tr>
<td>Create a definition for a column header.</td>
<td>DEFINE HEADER</td>
</tr>
<tr>
<td>Define a symbol that references a value that the data component supplies from the procedure or DATA step.</td>
<td>DYNAMIC</td>
</tr>
<tr>
<td>Define a symbol that references a macro variable. ODS will use the variable as a string. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.</td>
<td>MVAR</td>
</tr>
<tr>
<td>Define a symbol that references a macro variable. ODS will convert the variable’s value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.</td>
<td>NMVAR</td>
</tr>
<tr>
<td>Provide information about the column.</td>
<td>NOTES</td>
</tr>
<tr>
<td>Translate the specified values to other values.</td>
<td>TRANSLATE-INTO</td>
</tr>
<tr>
<td>End the definition.</td>
<td>END</td>
</tr>
</tbody>
</table>

**Required Arguments**

**column-path**  
specifies where to store the column definition. A **column-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 definition to the first template store that you can write to in the current path.  
**Restriction:** If the definition is nested inside another definition, **definition-path** must be a single-level name because the nested definition is stored where the original definition is stored.
Restriction: If you want to reference the definition that you are creating from another definition, do not nest the definition inside another one. For example, if you want to reference a column definition from multiple tables, do not define the column inside a table definition.

Options

STORE=libref.template-store

specifies the template store in which to store the definition. If the template store does not exist, it is created.

Restriction: If the definition is nested inside another definition, you cannot use the STORE= option for the nested definition because it is stored where the original definition is stored.

Restriction: The STORE= option does not become part of the definition.

Column Attributes

This section lists all the attributes that you can use in a column definition. For all attributes that support a value of ON, the following forms are equivalent:

ATTRIBUTE-NAME
ATTRIBUTE-NAME=ON

For all of the attributes that support a value of variable, variable can be any variable that you declare in the column definition 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 the following table:

Table 10.3 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 10.4 Column Attributes

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Influence the appearance of the cells</td>
<td>BLANK_DUPS</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>contents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether or not to suppress the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value of a variable from one row to the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next, if the value does not change based</td>
<td></td>
<td></td>
</tr>
<tr>
<td>on the formatted value of the variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Specify whether or not to suppress the value of a variable from one row to the next, if the value does not change based on the raw value of the variable.</td>
<td>BLANK_INTERNAL_DUPS</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Select the best format for a column of a table.</td>
<td>CHOOSE_FORMAT</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether or not to wrap the text in the current column.</td>
<td>FLOW</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the format for the column.</td>
<td>FORMAT</td>
<td>All</td>
</tr>
<tr>
<td>Specify the number of decimals for the column if it isn't specified with FORMAT= column attribute.</td>
<td>FORMAT_NDEC</td>
<td>All</td>
</tr>
<tr>
<td>Specify the format width for the column if it isn't specified with FORMAT= column attribute.</td>
<td>FORMAT_WIDTH</td>
<td>All</td>
</tr>
<tr>
<td>Supply a numeric value against which values in the column are compared to eliminate trivial values from printing.</td>
<td>FUZZ</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify the horizontal justification of the format field within the column (and for the column header if the definition for the header does not include JUST=).</td>
<td>JUST</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether to justify the format field within the column, or to justify the value within the column, without regard to the format field.</td>
<td>JUSTIFY</td>
<td>All destinations except LISTING behave as if JUSTIFY=ON.</td>
</tr>
<tr>
<td>When the text in the column uses more than one line, specify whether to try to divide the text equally among all lines or to maximize the amount of text in each line.</td>
<td>MAXIMIZE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not 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).</td>
<td>OVERLINE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to treat the text as preformatted text.</td>
<td>PREFORMATTED</td>
<td>HTML, MARKUP family, PRINTER family, and RTF</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Specify whether or not to print the column.</td>
<td>PRINT</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify a separator character to append to each value in the column.</td>
<td>SEPARATOR=</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the style element and style attributes to use for the column.</td>
<td>STYLE=</td>
<td>HTML, MARKUP family, PRINTER family, and RTF</td>
</tr>
<tr>
<td>Specify the split character for the data in the column.</td>
<td>TEXT_SPLIT=</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether or not 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).</td>
<td>UNDERLINE=</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the vertical justification for the column.</td>
<td>VJUST=</td>
<td>HTML, MARKUP family, PRINTER family, and RTF</td>
</tr>
<tr>
<td>Specify the width of the column in characters.</td>
<td>WIDTH=</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the maximum width for this column.</td>
<td>WIDTH_MAX=</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.</td>
<td>HEADER=</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether or not to print the column header.</td>
<td>PRINT_HEADERS</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Influence the relationship to other columns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether or not the column definition is generic; that is, whether or not it can be used by more than one variable.</td>
<td>GENERIC=</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether or not the column is an ID column.</td>
<td>ID</td>
<td>LISTING and PRINTER family</td>
</tr>
<tr>
<td>Specify whether or not to merge the current column with the column immediately to its right.</td>
<td>MERGE</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether or not to merge the current column with the column immediately to its left.</td>
<td>PRE_MERGE</td>
<td>All except OUTPUT</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>PRE_SPACE=</td>
<td>LISTING</td>
</tr>
</tbody>
</table>
### Task Attribute Valid Destinations

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the number of blank characters to leave between the current column and the column immediately to its right.</td>
<td>SPACE=</td>
<td>LISTING</td>
</tr>
<tr>
<td><strong>Influence the presentation of data panels</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Influence the place at which ODS splits a table when it creates multiple data panels.</td>
<td>GLUE=</td>
<td></td>
</tr>
<tr>
<td>Specify whether or not 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.</td>
<td>OPTIONAL</td>
<td>LISTING</td>
</tr>
<tr>
<td><strong>Other column attributes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify which format to use if both a column definition and a data component specify a format.</td>
<td>DATA_FORMAT_OVERRIDE</td>
<td>All</td>
</tr>
<tr>
<td>Specify the name of the column in the data component to associate with the current column.</td>
<td>DATANAME=</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</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether or not to include the column in an output data set.</td>
<td>DROP</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>Specify a label for the column.</td>
<td>LABEL=</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>Specify the column definition that the current definition inherits from.</td>
<td>PARENT=</td>
<td>All</td>
</tr>
<tr>
<td>Specify the name to use for the corresponding variable in an output data set.</td>
<td>VARNAME=</td>
<td>OUTPUT</td>
</tr>
</tbody>
</table>

**BLANK_DUPS<=ON | OFF | variable**
specifies whether or not to suppress the value of a variable from one row to the next, if the value does not change based on the formatted value of the variable.

**Default:** OFF

**Interaction:** If the CLASSLEVELS= table attribute is in effect, ODS ignores BLANK_DUPS=ON when any value changes in a preceding column that is also marked with BLANK_DUPS=ON.
ODS Destinations: All except OUTPUT. Note that when the PRINTER destination suppresses the value of a variable, it also suppresses the horizontal rule above the blank cell.

Featured in: Example 5 on page 539

**BLANK_INTERNAL_DUPS<=ON | OFF | variable>**
specifies whether or not to suppress the value of a variable from one row to the next, if the value does not change based on the raw value of the variable.

Default: OFF

Interaction: If the CLASSLEVELS= table attribute is in effect, ODS ignores BLANK_INTERNAL_DUPS=ON when any value changes in a preceding column that is also marked with BLANK_INTERNAL_DUPS=ON.

ODS Destinations: All except OUTPUT. Note that when the PRINTER destination suppresses the value of a variable, it also suppresses the horizontal rule above the blank cell.

**CHOOSE_FORMAT= COMPROMISE | MAX | MAX_ABS | MIN_MAX**
selects a format based on the actual values in the column of the table.

Default: If you omit the CHOOSE_FORMAT column attribute, then the default format is either determined by the data component or by other attributes.

Restriction: CHOOSE_FORMAT is not supported for computed columns because those columns' values are computed outside of the data object.

Tip: If you specify a small value for the FORMAT_WIDTH= option, then CHOOSE_FORMAT may create a dw.3 format.

ODS Destinations: All

See: For more information about column formats, see “How Are Values in Table Columns Formatted?” on page 514.

**COMPROMISE**
looks at all of the values in the column and selects a good compromise format to work well for most values, but extreme values may shift to BEST format.

Tip: FORMAT_NDEC=d specifies the precision in digits.

Tip: The FORMAT_WIDTH= option suggests a maximum width. The actual format width may be smaller or it may be larger.

**MAX**
selects a format based on the maximum value in the column. Values are all expected to be positive so no space is reserved for a minus sign.

Default: By default, FORMAT_WIDTH=10 and FORMAT_NDEC= is ignored.

**MAX_ABS**
selects a format based on the maximum absolute value in the column. The format reserves space for a minus sign whether it is needed or not.

**MIN_MAX**
selects a format based on the minimum and maximum value in the column. The format reserves space for a minus sign only where it is actually needed.

Interaction: If FORMAT_NDEC=d is specified, a maximum of d decimal places is used.

**DATA_FORMAT_OVERRIDE<=ON | OFF | variable>**
specifies which format to use if both a column definition and a data component specify a format.

Default: OFF

ODS Destinations: All
ON
  Uses the format in the data component.

OFF
  Uses the format in the column definition.

variable
  Uses the format of the specified variable.

```
DATANAME=column-name
```
specifies the name of the column in the data component to associate with the current column.

Default: By default, ODS associates the current column with a column of the same name in the data component.

ODS Destinations: All

```
DEF_SPLIT
```
specifies which special characters in headers for generic columns are to be used as split characters.

ODS Destinations: All

```
DROP<=ON | OFF | variable>
```
specifies whether or not to include the column in an output data set.

Default: OFF

ODS Destinations: OUTPUT

```
FLOW<=ON | OFF | variable>
```
specifies whether or not to wrap the text in the current column if it is too long to fit in the space that is provided.

Default: ON if the format width of the column is greater than the column width.
OFF if the format width of the column is not greater than the column width.

See also: MAXIMIZE= on page 384

ODS Destinations: LISTING

Note: The HTML and PRINTER destinations always wrap the text if it is too long to fit in the space that is provided.

```
FORMAT=format-name <format-width <decimal-width>> | variable
```
specifies the format for the column.

Default: If you do not specify the FORMAT= , PROC TEMPLATE uses the format that the data component provides. If the data component does not provide a format, PROC TEMPLATE uses

- BEST8. for integers
- 12.3 for doubles
- the length of the variable for character variables.

Restriction: If you specify a format width for a numeric column, then its value cannot exceed 32.

ODS Destinations: All

```
FORMAT_NDEC= number | variable
```
specifies the number of decimals for the column.

Default: the decimal width that is specified with the FORMAT= column attribute.

Range: Number is a whole number from 0 to 32

Interaction: If you specify a decimal width using both the FORMAT= and the FORMAT_NDEC= attributes, then PROC TEMPLATE uses the width that you specify with the FORMAT= attribute.
ODS Destinations: All

**FORMAT_WIDTH=** `positive-integer | variable`
specifies the format width for the column.

**Default:** If you omit the column attribute **FORMAT_WIDTH=**, then the format specified in the **FORMAT=** attribute is used.

**Range:** 1 to 32 for numeric variables; operating system limit for character variables

**Interaction:** If you specify a format width using both the **FORMAT=** and the **FORMAT_WIDTH=** attributes, then PROC TEMPLATE uses the width that you specify with the **FORMAT=** attribute.

**ODS Destinations:** All

**FUZZ=** `number | variable`
supplies a numeric value against which values in the column are compared to eliminate trivial values from printing. A number whose absolute value is less than or equal to the **FUZZ=** value is printed as 0. However, the real value of the number is used in any computations based on that number.

**Default:** the smallest representable floating-point number on the computer that you are using

**ODS Destinations:** All except OUTPUT

**GENERIC=** `<ON | OFF | variable>`
specifies whether or not the column definition can be used by more than one column. Generic columns are useful in tables with many similar columns. For example, the table definitions 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 definition the data component should use for each column.

**Default:** OFF

**ODS Destinations:** All except OUTPUT

**Featured in:** Example 3 on page 528 and Example 5 on page 539

**GLUE=** `integer | variable`
Influences the places at which ODS splits a table when it creates multiple data panels. ODS creates multiple data panels from a table that is too wide to fit in the allotted space. The higher the value of **GLUE=** is, the less likely it is that ODS will split the table between the current column and the column to its right.

**Default:** 1

**Range:** -1 to 327

**Tip:** A value of -1 forces the table to split between the current column and the column to its right.

**ODS Destinations:** LISTING, PRINTER family, and RTF

**HEADER=** `header-specification`
specifies the text for the column header. `header-specification` can be one of the following:

- `'text'`
  specifies the actual text of the header.
  **Requirement:** `text` must be enclosed by quotation marks.

- `header-name`
  specifies the name of a header definition to use. You create a header definition with the DEFINE HEADER statement (see “DEFINE HEADER Statement” on page 395). If `header-name` is a single-level name, the header definition must occur within the current column definition.
variable
specifies the name of a variable that you declare 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, you must use the LABEL= attribute.

Tip: The HEADER= option provides a simple way for you to specify the text of a column header. If you want to customize the header further, use the DEFINE HEADER statement with the appropriate header attributes. (See “DEFINE HEADER Statement” on page 395.)

Tip: You can use the split character in the text of the header to force the text to a new line.

See also: LABEL= on page 384 and TEXT_SPLIT= on page 387.

ODS Destinations:
All

Featured in: Example 3 on page 528 and Example 1 on page 342

ID<=ON | OFF | variable
specifies whether or not the column is an ID column. An ID column is repeated on each data panel. (ODS creates multiple data panels when a table is too wide to fit in the allotted space.)

Default: OFF

Tip: ODS treats all columns up to and including a column that is marked with ID=ON as ID columns.

ODS Destinations: LISTING and PRINTER family

Featured in: Example 3 on page 528

JUST=justification | variable
specifies the horizontal justification of the format field within the column (and of the header if the definition for the header does not include JUST=).

justification can be one of the following:

LEFT
specifies left justification.

Alias: L

RIGHT
specifies right justification.

Alias: R

CENTER
specifies center justification.

Alias: C

Default: LEFT for columns that contain character values; RIGHT for columns that contain numeric values.

Interaction: For the Listing destination, ODS justifies the format field within the column width. At times, you may need to specify the JUSTIFY= attribute to get the results that you want. See the discussion of JUSTIFY= on page 384.

See also: FORMAT= on page 381 and WIDTH= on page 388.

Main discussion: “How Are Values in Table Columns Justified?” on page 512
**ODS Destinations:** All except OUTPUT  
**Featured in:** Example 1 on page 515

**JUSTIFY<=ON | OFF | variable**
specifies whether to justify the format field within the column or to justify the value within the column without regard to the format field.

**Default:** OFF  
**Interaction:** JUSTIFY=ON can interfere with decimal alignment.  
**Tip:** If you translate numeric data to character data, you may need to use JUSTIFY= to align the data.  
**Main discussion:** “How Are Values in Table Columns Justified?” on page 512  
**Featured in:** Example 5 on page 539  
**ODS Destinations:** All destinations except LISTING behave as if JUSTIFY=ON

**LABEL='text' | variable**
specifies a label for the column in the output data set.

**Default:** If you do not specify 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.  
**ODS Destinations:** OUTPUT  
**Tip:** If the Output destination is open, LABEL= 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 might 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 would result in 33% of the text on each line.

MAXIMIZE=ON may write lines of text that vary greatly in length. MAXIMIZE=OFF may result in using less than the full column width.

**Default:** OFF  
**Interaction:** This attribute is effective only if the column is defined with FLOW=ON (see the discussion of FLOW= on page 381).  
**ODS Destinations:** LISTING

**MERGE<=ON | OFF | variable**
specifies whether or not to merge the current column with the column immediately to its right. When you set MERGE=ON for the current column, the data in each row of the column is merged with the data in the same row of the next column. ODS applies the format, justification, spacing, and prespacing attributes to each column independently. Then, it concatenates the columns. Finally, it applies to the concatenated data all the remaining attributes that are specified on the column that does not have MERGE= set.

**Default:** OFF  
**Restriction:** You cannot use both MERGE=ON and PRE_MERGE=ON in the same column definition. 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.

**See also:** PRE_MERGE= on page 385  
**ODS Destinations:** All except OUTPUT
OPTIONAL<=ON | OFF | variable >=
specifies whether or not to delete the current column from the output object if doing so enables all the remaining columns to fit in the space that is provided without splitting the table into multiple data panels.

Default: OFF

Interaction: If multiple column definitions contain OPTIONAL=ON, PROC TEMPLATE includes either all or none of these columns in the output object.

ODS Destinations: LISTING

OVERLINE<=ON | OFF | variable >=
specifies whether or not 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). PROC TEMPLATE uses the second formatting character to draw the line. (See the discussion of FORMCHAR= on page 417.)

Default: OFF

ODS Destinations: LISTING

PARENT=column-path
specifies the column definition that the current definition inherits from. A column-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 definition inherits from the specified column in the first template store that you can read from in the current path.

When you specify a parent, all the attributes and statements that are specified in the parent’s definition are used in the current definition unless the current definition specifically overrides them.

ODS Destinations: All

PREFORMATTED<=ON | OFF | variable >=
specifies whether or not 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.

ODS Destinations: HTML, MARKUP family, PRINTER family, and RTF

PRE_MERGE<=ON | OFF | variable >=
specifies whether or not 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 definition. 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.

See also: MERGE= on page 384

ODS Destinations: All except OUTPUT
**PRE_SPACE=**\(\text{non-negative-integer}\)
specifies the number of blank characters to leave between the current column and the column immediately to its left.

**Default:** A value in the range that is bounded by the COL_SPACE_MIN and COL_SPACE_MAX table attributes.

**Interaction:** If PRE_SPACE= and SPACE= are specified for the same intercolumn space, ODS honors PRE_SPACE=.

**See also:** SPACE= on page 386, COL_SPACE_MIN= on page 416, and COL_SPACE_MAX= on page 416

**ODS Destinations:** LISTING

**PRINT<ON | OFF | variable>**
specifies whether or not to print the column.

**Default:** ON

**See also:** OPTIONAL= on page 385 and DROP= on page 381

**ODS Destinations:** All except OUTPUT

**PRINT_HEADERS<=ON | OFF | variable>**
specifies whether or not to print the column header and any underlining and overlining.

**Default:** ON

**See also:** UNDERLINE= on page 387 and OVERLINE= on page 385

**ODS Destinations:** All except OUTPUT

**SEPARATOR='character' | variable**
specifies a separator character to append to each value in the column.

**Default:** None

**Restriction:** The SEPARATOR= column attribute is valid only for character variables.

**Tip:** To specify a hexadecimal character as the separator character, put an x after the closing quote. For example, the following option assigns the hexadecimal character 2D as the separator character:

```
separator='2D'x
```

**ODS Destinations:** LISTING

**SPACE=**\(\text{positive-integer} \ | \ \text{variable}\)
specifies the number of blank characters to leave between the current column and the column immediately to its right.

**Default:** A value in the range that is bounded by the COL_SPACE_MIN and COL_SPACE_MAX table attributes.

**Interaction:** If PRE_SPACE= and SPACE= are specified for the same intercolumn space, ODS honors PRE_SPACE=.

**See also:** PRE_SPACE= on page 386, COL_SPACE_MIN= on page 416, and COL_SPACE_MAX= on page 416

**ODS Destinations:** LISTING

**STYLE=**\(<\text{style-element-name}>\{\text{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 definition that is registered with the Output Delivery System. SAS provides some style definitions. You can create your own style definitions with PROC TEMPLATE (see “DEFINE STYLE Statement” on page 288). 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 be most likely to use with the STYLE= column attribute are

- 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 on viewing a style definition so that you can see the style elements that are available, see “Viewing the Contents of a Style Definition” on page 319. For information about the default style definition that ODS uses, see “The Default Style Definition for HTML and Markup Languages” on page 320.

style-element-name can be either the name of a style element or a variable whose value is a style element.

**Default:** data

**style-attribute-specification** describes the style attribute to change. Each style-attribute-specification has this general form:

style-attribute-name=style-attribute-value

For information on the style attributes that you can specify, see “Style Attributes and Their Values” on page 292.

**ODS Destinations:** HTML, MARKUP family, PRINTER family, and RTF

**Featured in:** Example 3 on page 528

**Tip:** If you use the STYLE= attribute inside a quoted string, then you must 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 your quoted strings.

**TEXT_SPLIT=’character’ | variable** specifies the split character for the data in the column. PROC TEMPLATE breaks a value in the column 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

**ODS Destinations:** All except OUTPUT

**UNDERLINE=ON | OFF | variable** specifies whether or not 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). PROC TEMPLATE uses the second formatting character to draw the line.

**Default:** OFF
**Main discussion:** See the discussion of FORMCHAR= on page 417.

**ODS Destinations:** LISTING

VARNAME=variable-name | variable

specifies the name to use for the corresponding variable in an output data set.

**Default:** If you do not specify VARNAME=, PROC TEMPLATE uses the value of the DATANAME= attribute. If you do not specify DATANAME=, PROC TEMPLATE uses the name of the column.

**Tip:** If you use VARNAME= to specify the same name for different columns, a number is appended to the name each time that the name is used.

**ODS Destinations:** OUTPUT

VJUST=justification | variable

Specifies the vertical justification for the column. justification can be 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 for the Printer destination; CENTER for the HTML destination

**ODS Destinations:** HTML, MARKUP family, PRINTER family, and RTF

**Featured in:** Example 3 on page 528

WIDTH=positive-integer | variable

specifies the width of the column in characters.

**Default:** If you do not specify a width, PROC TEMPLATE uses the format width. If the column has no format associated with it, PROC TEMPLATE uses a width of

- 8 for integers
- 12 for doubles
- data length for character variables.

**Interaction:** The length of the column header can influence the width of the column.

**See also:** WIDTH_MAX on page 388 and WIDTH= on page 406.

**ODS Destinations:** LISTING

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

**ODS Destinations:** LISTING

---

**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, flyover) of individual cells

**Featured in:** Example 5 on page 539
**CELLSTYLE** `expression-1 AS <style-element-name><[style-attribute-specification(s)]> AS <expression-n AS <style-element-name><[style-attribute-specification(s)]>>;`

### Required Arguments

**expression**

is an expression that is evaluated for each cell in the column. It can be any expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in SAS Language Reference: Dictionary. Use _VAL_ to represent the value of the current column. You may also reference symbols that you declared in a DYNAMIC, MVAR, or NMVAR statement in the definition.

If `expression` resolves to TRUE (a non-zero value), the style element that is specified is used for the current cell. If `expression` is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

**Restriction:** You can not reference the values of other columns in `expression`.

**Tip:** Using an expression of 1 as the last expression in the CELLSTYLE-AS statement sets the style element for any cells that did not meet an earlier condition.

### Options

**Note:** Neither `style-attribute-specification` nor `style-element-name` is required. However, you must use at least one of them.

**style-attribute-specification**

describes a style attribute to set. Each `style-attribute-specification` has this general form:

```
style-attribute-name=style-attribute-value
```

For information on the style attributes that you can set in a column definition, see “Style Attributes and Their Values” on page 292.

**Default:** If you don’t specify any style attributes to modify, ODS uses the unmodified `style-element-name`.

**style-element-name**

is the name of the style element that is used to display the data in the column. The style element must be part of a style definition that is registered with the Output Delivery System. SAS provides some style definitions. You can create your own style definitions by using PROC TEMPLATE (see “DEFINE STYLE Statement” on page 288). 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 be most likely to use with the CELLSTYLE-AS statement in a column definition are the following.

- data
- datafixed
- dataempty
- dataemphasis
- dataemphasisfixed
- datastrong
The style element provides the basis for displaying the column. Additional style attributes that you provide can modify the display.

**Default:** data

**See also:** “Viewing the Contents of a Style Definition” on page 319.

**See also:** “The Default Style Definition for HTML and Markup Languages” on page 320.

### COMPUTE AS Statement

Computes values for a column that is not in the data component, or modifies the values of a column that is in the data component

**COMPUTE AS** *expression*;

#### Required Arguments

**expression**

is an expression that assigns a value to each table cell in the column. It can be any expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in *SAS Language Reference: Dictionary*.

To reference another column in a COMPUTE-AS statement, use the name of the column. You can also reference symbols that you declared in a DYNAMIC, MVAR, or NMVAR statement in the current definition. In addition, if the column has values in the data component, you can reference the column itself in the expression. However, if you are creating a column that does not exist in the data component, you cannot reference the column in the expression because there is no underlying value to use.

For example, the following 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;
```

**Tip:** The COMPUTE AS statement can alter values in an output object. None of the definitions that SAS provides modifies any values. If you want to determine if a definition was provided by SAS, use the “ODS VERIFY Statement” on page 205. If the definition is not from SAS, the ODS VERIFY statement returns a warning when it runs the SAS program that uses the definition. If you receive such a warning, you can use the SOURCE statement to look at the definition and determine if the COMPUTE AS statement is used to alter values. (See “SOURCE Statement” on page 277.)

**Tip:** Because you can use column names in expression, _VAL_ is not recognized as an alias for the current column.
DEFINE HEADER Statement

Creates a definition for a header inside a column definition

Main discussion: “DEFINE HEADER Statement” on page 395

```
DEFINE HEADER definition-name;
  statements-and-attributes
END;
```

Required Arguments

definition-name
  specifies the name of the new header.
  Restriction: definition-name must be a single-level name.
  Note: If you want to reference the header definition that you are creating from another definition, you must create it outside the column definition.△

statements-and-attributes
  specify the statements and header attributes that you can use to define a header inside a column.
  See: “DEFINE HEADER Statement” on page 395

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step

Scope: You can use the DYNAMIC statement in the definition of a table, column, header, or footer. A dynamic variable that is defined in a definition is available to that definition and to all the definitions that it contains.

Featured in: Example 1 on page 342 and Example 2 on page 348

```
DYNAMIC variable-1 <’text-1’> <… variable-n <’text-n’>>;
```

Required Arguments

variable
  Names a variable that the data component supplies. ODS resolves the value of the variable when it binds the definition and the data component.
  Tip: Dynamic variables are most useful to the authors of SAS procedures and to DATA step programmers.
Options

text

is text that you can place in the definition to explain the dynamic variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

MVAR Statement

Defines a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.

Scope: You can use the MVAR statement in the definition of a table, column, header, or footer. A macro variable that is defined in a definition is available to that definition and to all the definitions that it contains.

Featured in: Example 3 on page 528 and Example 1 on page 342

MVAR variable-1 <’text-1’> <… variable-n <’text-n’>>;

Required Arguments

variable

Names a macro variable to reference in the definition. ODS will use the value of the macro variable as a string. ODS does not resolve the value of the macro variable until it binds the definition and the data component.

Tip: You must declare macro variables this way in a definition. For example, to use the automatic macro variable SYSDATE9 in a definition, declare it in an MVAR statement and reference it as SYSDATE9, without an ampersand, in your PROC TEMPLATE step. If you use the ampersand, the macro variable resolves when the definition is compiled instead of when ODS binds the definition to the data component.

Options

text

is text that you can place in the definition to explain the macro variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.
NMVAR Statement

Defines a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.

Scope: You can use the NMVAR statement in the definition of a table, column, header, or footer. A macro variable that is defined in a definition is available to that definition and to all the definitions that it contains.

Featured in: Example 5 on page 539

NMVAR variable-1 <'text-1'> <... variable-n <'text-n'>>;

Required Arguments

variable

Names a macro variable to reference in the definition. ODS will convert the variable’s value to a number (stored as a double) before using it. ODS does not resolve the macro variable until it binds the definition and the data component.

Tip: You must declare macro variables this way in a definition. 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 definition is compiled instead of when ODS binds the definition to the data component.

Options

text

is text that you can place in the definition to explain the macro variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES Statement

Provides information about the column

Tip: The NOTES statement becomes part of the compiled column definition, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES 'text';
Required Arguments

text

provides information about the column.

---

**TRANSLATE-INTO Statement**

Translates the specified values to other values

**TRANSLATE** `expression-1` **INTO** `expression-2` `<...` `expression-n` **INTO** `expression-m` `>;

Required Arguments

**expression-1**

is an expression that is evaluated for each table cell in the column. It can be any expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in *SAS Language Reference: Dictionary*. Use `_VAL_` to represent the value of the current column. You may also reference symbols that you declared in a DYNAMIC, MVAR, or NVAR statement in the table definition.

If `expression-1` resolves to TRUE (a non-zero value), the translation that is specified is used for the current cell. If `expression-1` is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

**Restriction:** You may not 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.

**expression-2**

is an expression that specifies the value to use in the cell in place of the variable’s actual value. It can be any expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in *SAS Language Reference: Dictionary*. Use `_VAL_` to represent the value of the current column. You may also reference symbols that you declared in a DYNAMIC, MVAR, or NVAR statement in the table definition.

**Restriction:** `expression-2` must resolve to a character value, not a numeric value.

**Restriction:** You may not reference the values of other columns in `expression-2`.

**Tip:** When you translate a numeric value to a character value, the column definition does not try to apply the numeric format that is associated with the column. Instead, it simply writes the character value into the format field, starting at the left. If you want the value to be right-justified, use the JUSTIFY=ON attribute.

**See also:** JUSTIFY= on page 384.
END Statement

Ends the definition

END;

DEFINE FOOTER Statement

Creates a definition for a table footer

Requirement: An END statement must be the last statement in the definition.
Featured in: Example 3 on page 528 and Example 1 on page 342
See: “DEFINE HEADER Statement” on page 395

DEFINE FOOTER footer-path< / STORE=libref.template-store>;<footer-attribute-1; <… footer-attribute-n; >>
  DYNAMIC variable-1 <'text-1'> <… variable-n <'text-n'>>; MVAR variable-1 <'text-1'> <… variable-n <'text-n'>>; NMVAR variable-1 <'text-1'> <… variable-n <'text-n'>>; NOTES 'text'; TEXT footer-specification;
TEXT2 footer-specification;
TEXT3 footer-specification;
END;

The substatements in 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 “DEFINE HEADER Statement” on page 395.

DEFINE HEADER Statement

Creates a definition for a table header

Requirement: An END statement must be the last statement in the definition.
Featured in: Example 3 on page 528
**DEFINE HEADER** header-path </ STORE=libref.template-store>;  
<header-attribute-1; <… header-attribute-n; >>  
**DYNAMIC** variable-1 '<text-1'> <… variable-n '<text-n'>>;  
**MVAR** variable-1 '<text-1'> <… variable-n '<text-n'>>;  
**NMVAR** variable-1 '<text-1'> <… variable-n '<text-n'>>;  
**NOTES** 'text';  
**TEXT** header-specification;  
**TEXT2** header-specification;  
**TEXT3** header-specification;  
**END**;

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set one or more header attributes.</td>
<td>header-attribute(s)</td>
</tr>
<tr>
<td>Define a symbol that references a value that the data component supplies from the procedure or DATA step.</td>
<td>DYNAMIC</td>
</tr>
<tr>
<td>Define a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.</td>
<td>MVAR</td>
</tr>
<tr>
<td>Define a symbol that references a macro variable. ODS will convert the variable’s value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.</td>
<td>NMVAR</td>
</tr>
<tr>
<td>Provide information about the table.</td>
<td>NOTES</td>
</tr>
<tr>
<td>Specify the text of the header.</td>
<td>TEXT</td>
</tr>
<tr>
<td>Specify an alternative header to use in the listing output if the header that is provided by the TEXT statement is too long.</td>
<td>TEXT2</td>
</tr>
<tr>
<td>Specify an alternative header to use in the Listing output if the header that is provided by the TEXT2 statement is too long.</td>
<td>TEXT3</td>
</tr>
<tr>
<td>End the header definition.</td>
<td>END</td>
</tr>
</tbody>
</table>

**Required Arguments**

**header-path**  
specifies where to store the header definition. 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 definition to the first template store that you can write to in the current path.

**Restriction:** If the definition is nested inside of another definition, *definition-path* must be a single-level name.

**Restriction:** If you want to reference the definition that you are creating from another definition, then do not nest the definition inside another definition. For example, if you want to reference a header definition from multiple columns, do not define the header inside a column definition.

**Options**

**STORE=**<code>libref.template-store</code>

specifies the template store in which to store the definition. If the template store does not exist, it is created.

**Restriction:** If the definition is nested inside another definition, you cannot use the STORE= option for the nested definition because it is stored where the original definition is stored.

**Restriction:** The STORE= option does not become part of the definition.

**Header Attributes**

This section lists all the attributes that you can use in a header definition. A column header spans a single column. A spanning header spans multiple columns. These two kinds of headers are defined in the same way except that a spanning header uses the START= or the END= attribute, or both.

For all attributes that support a value of ON, the following forms are equivalent:

```
ATTRIBUTE-NAME = ON
```

For all of the attributes that support a value of *variable*, *variable* can be any variable that you declare in the table definition 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 the following table:

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>rookies</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>rookies</td>
<td><em>NO</em></td>
</tr>
</tbody>
</table>
Table 10.7  Header Attributes

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Influence the appearance of the contents of the header</td>
<td>DEF_SPLIT</td>
<td>All</td>
</tr>
<tr>
<td>Specify that special characters in headers for generic columns are to be used as split characters.</td>
<td>FORCE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to try to expand the column width to accommodate the longest word in the column header.</td>
<td>JUST=</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify the horizontal justification for the column header.</td>
<td>MAXIMIZE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether to try to divide the text equally among all lines or to maximize the amount of text in each line when the text in the header uses more than one line.</td>
<td>OVERLINE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to draw a continuous line above the header.</td>
<td>PRINT</td>
<td>All</td>
</tr>
<tr>
<td>Specify whether or not to treat the text as preformatted text.</td>
<td>SPACE=</td>
<td>LISTING</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>SPLIT=</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify the split character for the header.</td>
<td>STYLE=</td>
<td>HTML, PRINTER family, and RTF</td>
</tr>
<tr>
<td>Specify the style element and any changes to its attributes to use for the header.</td>
<td>TRUNCATE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to start a new header line in the middle of a word.</td>
<td>UNDERLINE</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to draw a continuous line underneath the header.</td>
<td>VJUST=</td>
<td>HTML, MARKUP family, PRINTER, family, and RTF</td>
</tr>
<tr>
<td>Specify vertical justification for the header.</td>
<td>WIDTH=</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify the width of the header in characters.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TASK | ATTRIBUTE | VALID DESTINATIONS
--- | --- | ---
**Influence the content of the header**
Specify a character to use to expand the header to fill the space over the column or columns that the header spans. | EXPAND= | LISTING
Specify whether or not to repeat the text of the header until the space that is allotted for the header is filled. | REPEAT | LISTING
**Influence the placement of the header**
Specify the last column that a spanning header covers. | END= | All except OUTPUT
Specify the first column that a spanning header covers. | START= | All except OUTPUT
Specify whether or not to expand the header to reach the sides of the page. | EXPAND_PAGE | LISTING
Specify whether or not a spanning header appears only on the first data panel if the table is too wide to fit in the space that is provided. | FIRST_PANEL | LISTING, PRINTER family, and RTF
Specify whether or not a table footer appears only on the last data panel if the table is too wide to fit in the space that is provided. | LAST_PANEL | LISTING, PRINTER family, and RTF
Specify whether or not to extend the text of the header into the header space of adjacent columns. | SPILL_ADJ | LISTING
Specify whether or not to extend the text of the header into the adjacent margin. | SPILL_MARGIN | LISTING
**Other header attributes**
Specify an abbreviation for the header.* | ABBR= | MARKUP
Specify an acronym for the header.* | ACRONYM= | MARKUP
Specify an alternate description for the header.* | ALT= | MARKUP
Specify whether or not multiple columns can use the header. | GENERIC | All except OUTPUT
<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a long description for the header.*</td>
<td>LONGDESC=</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Specify the header definition that the current definition inherits from.</td>
<td>PARENT=</td>
<td>All</td>
</tr>
</tbody>
</table>

* SAS includes these accessibility and compatibility features that improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

**ABBR= '*text'**

specifies an abbreviation for the header.

**Requirement:** The text must be enclosed with quotation marks.

**ODS Destinations:** MARKUP

**Note:** SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

**ACRONYM= '*text'**

specifies an acronym for the header.

**Requirement:** The text must be enclosed with quotation marks.

**ODS Destinations:** MARKUP

**Note:** SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

**ALT= '*text'**

specifies an alternate description of the header.

**Requirement:** The text must be enclosed with quotation marks.

**ODS Destinations:** MARKUP

**Note:** SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

**DEF_SPLIT**

specifies which special characters in headers for generic columns are to be used as split characters.

**ODS Destinations:** All

**END=column-name | variable**

specifies the last column that a spanning header covers.

**Default:** the last column

**See also:** START= on page 404

**ODS Destinations:** All except OUTPUT

**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.
**DEFINE HEADER Statement**

**Default:** none

**Interaction:** If you specify both the REPEAT=ON and EXPAND=ON attributes, then PROC TEMPLATE uses the EXPAND= attribute.

**See also:** REPEAT= on page 403

**Tip:** If the string or the variable that you specify contains more than one character, then PROC TEMPLATE uses only the first character.

**See also:** EXPAND_PAGE= on page 401

**ODS Destinations:** LISTING

**EXPAND_PAGE=**< OFF | variable>

specifies whether or not to expand the header to reach the sides of the page.

**Default:** OFF

**See also:** EXPAND= on page 400

**ODS Destinations:** LISTING

**FIRST_PANEL=**< ON | OFF | variable>

specifies whether or not a spanning header appears only on the first data panel if the table is too wide to fit in the space that is provided.

**Default:** OFF

**Restriction:** Applies only to headers, not to footers

**See also:** LAST_PANEL= on page 402

**ODS Destinations:** LISTING, PRINTER family, and RTF

**FORCE=**< ON | OFF | variable>

specifies whether or not to try to expand the column width to accommodate the longest word in the column header. The column width can be anything between the values for the WIDTH= and WIDTH_MAX= column attributes.

**Default:** ON

**See also:** WIDTH= on page 406 and WIDTH_MAX= on page 388

**ODS Destinations:** LISTING

**GENERIC=**< ON | OFF | variable>

specifies whether or not multiple columns can use the header.

**Default:** OFF

**Restriction:** This attribute is primarily for writers of SAS procedures and for DATA step programmers.

**ODS Destinations:** All except OUTPUT

**JUST=justification | variable**

specifies the horizontal justification for the column header, where justification can be 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
ODS Destinations: All except OUTPUT

Featured in: Example 1 on page 515

LAST_PANEL<= ON | OFF | variable>
specifies whether or not a table footer appears only on the last data panel if the table is too wide to fit in the space that is provided.

Default: OFF

Restriction: Applies only to footers, not to headers

See also: FIRST_PANEL on page 401

ODS Destinations: LISTING, PRINTER family, and RTF

LONGDESC= 'string'
specifies the long description of the header.

Requirement: The text must be enclosed with quotation marks.

ODS Destinations: MARKUP

Note: SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

MAXIMIZE<=ON | OFF | variable>
specifies whether to try to divide the text equally among all lines or to maximize the amount of text in each line when the text in the header uses more than one line. For example, if the text spans three lines, MAXIMIZE=ON might 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 would may result in 33% of the text on each line. MAXIMIZE=ON may write lines of text that vary greatly in length. MAXIMIZE=OFF may result in using less than the full column width.

Default: OFF

ODS Destinations: LISTING

OVERLINE<=ON | OFF | variable>
specifies whether or not to draw a continuous line above the header. PROC TEMPLATE uses the second formatting character to draw the line. (See the discussion of FORMCHAR= on page 417.)

Default: OFF

ODS Destinations: LISTING

PARENT=header-path
specifies the header definition that the current definition 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 definition inherits from the specified header definition in the first template store that you can read from in the current path.

When you specify a parent, all of the attributes and statements that are specified in the parent’s definition are used in the current definition unless the current definition specifically overrides them.

ODS Destinations: All

PREFORMATTED<=ON | OFF | variable>
specifies whether or not to treat the text as preformatted text. When text is preformatted, ODS honors line breaks as well as leading, trailing, and internal spaces. It also displays the text in a monospace font.

Default: OFF
**Interaction:** When PREFORMATTED=ON, and you are defining a table header or a footer, ODS uses the `headerfixed` or the `footerfixed` style element unless you specify another style element with the `STYLE=` column attribute.

When PREFORMATTED=ON, and you are defining a column header, ODS uses the `rowheaderfixed` style element unless you specify another style element with the `STYLE=` column attribute.

**ODS Destinations:** HTML, MARKUP family, PRINTER family, and RTF

**PRINT<=ON | OFF | variable>**
specifies whether or not to print the header.

**Default:** ON

**Tip:** When PRINT=ON, the column header becomes the label of the corresponding variable in any output data sets that the Output Destination creates if neither the column definition nor the data component provides a label.

**ODS Destinations:** All

**REPEAT<=ON | OFF | variable>**
specifies whether or not to repeat the text of the header until the space that is allotted for the header is filled.

**Default:** OFF

**Interaction:** If you specify both the REPEAT=ON and EXPAND=ON attributes, then PROC TEMPLATE uses the EXPAND= attribute.

**See also:** EXPAND= on page 400

**ODS Destinations:** LISTING

**SPACE=positive-integer | variable**
specifies the number of blank lines to place between the current header and the next header or between the current footer and the previous footer.

**Default:** 0

**Tip:** A row of underlining or overlining is considered a header or a footer.

**ODS Destinations:** LISTING

**Featured in:** Example 3 on page 528

**SPILL_ADJ<=ON | OFF | variable>**
specifies whether or not 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 PROC TEMPLATE uses only one of these attributes. `FORCE=` takes precedence over the other three attributes, followed by `SPILL_MARGIN=`, `SPILL_ADJ=`, and `TRUNCATE=`.

**See also:** `FORCE=` on page 401, `SPILL_MARGIN=` on page 403, and `TRUNCATE=` on page 405

**ODS Destinations:** LISTING

**SPILL_MARGIN<=ON | OFF | variable>**
specifies whether or not 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:** `FORCE=`, `SPILL_MARGIN=`, `SPILL_ADJ=`, and `TRUNCATE=` attributes are mutually exclusive. If you specify more than one of these attributes,
then PROC TEMPLATE uses only one of these attributes. FORCE= takes precedence over the other three attributes, followed by SPILL_Margin=, SPILL_ADJ=, and TRUNCATE=.

**See also:** FORCE= on page 401, SPILL_ADJ= on page 403, and TRUNCATE= on page 405

**ODS Destinations:** LISTING

**SPLIT='character' | variable**
specifies the split character for the header. PROC TEMPLATE starts a new line when it reaches that character and continues the header on the next line. The split character itself is not part of the header although each occurrence of the split character counts toward the maximum length for a label.

**Tip:** The first character in a header is automatically treated as a split character if it is not one of the following:

- an alphanumeric character
- a blank
- an underscore (_)
- a hyphen (-).

**ODS Destinations:** All except OUTPUT

**START=column-name | variable**
specifies the first column that a spanning header covers.

**Default:** the first column

**See also:** END= on page 400

**ODS Destinations:** All except OUTPUT

**STYLE=<style-element-name><[style-attribute-specification(s)]>**
specifies the style element and any changes to its attributes to use for the header.

**Requirement:** The STYLE= option requires either a style-attribute-specification or a style-element-name.

**Tip:** You can use braces ({ and }) instead of square brackets ([ and ]).

**Tip:** If you use the STYLE= attribute inside a quoted string, then you must 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 your quoted strings.

**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 definition that is registered with the Output Delivery System. SAS provides some style definitions. You can create your own style definitions by using PROC TEMPLATE (see “DEFINE STYLE Statement” on page 288). 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
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.

**Default:** header

**See also:** “Viewing the Contents of a Style Definition” on page 319.

**See also:** “The Default Style Definition for HTML and Markup Languages” on page 320.

**style-attribute-specification**

describes the style attribute to change. Each *style-attribute-specification* has this general form:

```
style-attribute-name=style-attribute-value
```

**ODS destinations:** HTML, PRINTER family, and RTF

**Featured in:** Example 1 on page 515 and Example 3 on page 528

**See also:** “Style Attributes and Their Values” on page 292

**TRUNCATE<=ON | OFF | variable>**
specifies whether or not to start a new header line in the middle of a word.

**ON**
starts a new line of the header when the text fills the specified column width.

**OFF**
extends the width of the column to accommodate the longest word in the column header, if possible.

**Note:** TRUNCATE=OFF is the same as FORCE=ON.

**Default:** OFF
Interaction: If you specify FORCE=, SPILL_MARGIN=, or SPILL_ADJ=, then the TRUNCATE= attribute is ignored.

See also: FORCE= on page 401, SPILL_MARGIN= on page 403, and SPILL_ADJ= on page 403

ODS Destinations: LISTING

UNDERLINE<ON | OFF | variable>
specifies whether or not to draw a continuous line below the header. PROC TEMPLATE uses the second formatting character to draw the line.

Default: OFF

Main discussion: See the discussion of FORMCHAR= on page 417.

ODS Destinations: LISTING

VJUST=justification | variable
Specifies vertical justification for the header. justification can be one of the following:

TOP
places the header as high as possible.

Alias: T

CENTER
centers the header vertically.

Alias: C

BOTTOM
places the header as low as possible.

Alias: B

Default: BOTTOM

ODS Destinations: HTML and PRINTER family

WIDTH=positive-integer | variable
specifies the width of the header in characters.

Default: If you do not specify a width, PROC TEMPLATE uses the column width.

Tip: If you want a vertical header, specify a width of 1.

ODS Destinations: LISTING

---

**DYNAMIC Statement**

Defines a symbol that references a value that the data component supplies from the procedure or DATA step.

Scope: You can use the DYNAMIC statement in the definition of a table, column, header, or footer. A dynamic variable that is defined in a definition is available to that definition and to all the definitions that it contains.

Featured in: Example 1 on page 342 and Example 2 on page 348

**DYNAMIC variable-1 <text-1> <... variable-n <text-n>>;**
Required Arguments

variable
Names a variable that the data component supplies. ODS resolves the value of the variable when it binds the definition and the data component.

Tip: Dynamic variables are most useful to the authors of SAS procedures and to DATA step programmers.

Options

text
is text that you can place in the definition to explain the dynamic variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

MVAR Statement

Defines a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object

Scope: You can use the MVAR statement in the definition of a table, column, header, or footer. A macro variable that is defined in a definition is available to that definition and to all the definitions that it contains.

Featured in: Example 3 on page 528 and Example 1 on page 342

MVAR variable-1 <'text-1'> <... variable-n <'text-n'>>;

Required Arguments

variable
Names a macro variable to reference in the definition. ODS will use the value of the macro variable as a string. ODS does not resolve the value of the macro variable until it binds the definition and the data component.

Tip: You must declare macro variables this way in a definition. For example, to use the automatic macro variable SYSDATE9 in a definition, declare it in an MVAR statement and reference it as SYSDATE9, without an ampersand, in your PROC TEMPLATE step. If you use the ampersand, the macro variable resolves when the definition is compiled instead of when ODS binds the definition to the data component.

Options

text
is text that you can place in the definition to explain the macro variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.
**NMVAR Statement**

Defines a symbol that references a macro variable. ODS will convert the variable’s value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.

**Scope:** You can use the NMVAR statement in the definition of a table, column, header, or footer. A macro variable that is defined in a definition is available to that definition and to all the definitions that it contains.

**Featured in:** Example 5 on page 539

```
NMVAR variable-1 '<text-1>' <... variable-n '<text-n'>>; 
```

**Required Arguments**

*variable*

Names a macro variable to reference in the definition. ODS will convert the variable’s value to a number (stored as a double) before using it. ODS does not resolve the macro variable until it binds the definition and the data component.

**Tip:** You must declare macro variables this way in a definition. 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 definition is compiled instead of when ODS binds the definition to the data component.

**Options**

*text*

is text that you can place in the definition to explain the macro variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

**NOTES Statement**

Provides information about the header

**Tip:** The NOTES statement becomes part of the compiled header definition, which you can view with the SOURCE statement, whereas SAS comments do not.

```
NOTES 'text';
```
Required Arguments

`text`

provides information about the header.

TEXT Statement

Specifies the text of the header or the label of a variable in an output data set

Featured in: Example 3 on page 528

```sql
TEXT header-specification(s);
```

Required Arguments

`header-specification(s)`

specifies the text of the header. Each `header-specification` can be one of the following:

`_LABEL_`

uses the label of the object that the header applies to as the text of the header. For example, if the header is for a column, `_LABEL_` specifies the label for the variable that is associated with the column. If the header is for a table, `_LABEL_` specifies the label for the data set that is associated with the table.

`text-specification(s)`

specifies the text to use in the header. Each `text-specification` can be one of the following:

- a quoted string
- a variable, followed by an optional format. The variable can be any variable that is declared in a DYNAMIC, MVAR, or NMVAR statement.

Note: If the first character in a quoted string is neither a blank character nor an alphanumeric character, and SPLIT is not in effect, the TEXT statement treats that character as the split character. (See the discussion of SPLIT= on page 404.)

Default: If you do not use a TEXT statement, the text of the header is the label of the object that the header applies to.

Tip: If the quoted string is a blank and it is the only item in the header specification, the header is a blank line.

Featured in: Example 3 on page 528

TEXT2 Statement

Provides an alternative header to use in the Listing output if the header that is provided by the TEXT statement is too long

See: “TEXT Statement” on page 409
TEXT3 Statement

Provides an alternative header to use in the Listing output if the header that is provided by the TEXT2 statement is too long

See: “TEXT Statement” on page 409

DEFINE TABLE Statement

Creates a table definition

Requirement: An END statement must be the last statement in the definition.

Interaction: A table definition can contain one or more column, header, or footer definitions.

Featured in: Example 3 on page 528 and Example 5 on page 539

Table 10.8 DEFINE TABLE Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set one or more table attributes.</td>
<td>table-attribute(s)</td>
</tr>
<tr>
<td>Set the style element of the cells in the table that contain numeric variables according to the values of the variables.</td>
<td>CELLSTYLE-AS</td>
</tr>
<tr>
<td>Task</td>
<td>Statement</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>Declare a symbol as a column in the table and specify the order of the columns.</td>
<td>COLUMN</td>
</tr>
<tr>
<td>Create a definition for a column, header, or footer.</td>
<td>DEFINE</td>
</tr>
<tr>
<td>Define a symbol that references a value that the data component supplies from the procedure or DATA step.</td>
<td>DYNAMIC</td>
</tr>
<tr>
<td>Declare a symbol as a footer in the table and specify the order of the footers.</td>
<td>FOOTER</td>
</tr>
<tr>
<td>Declare a symbol as a header in the table and specify the order of the headers.</td>
<td>HEADER</td>
</tr>
<tr>
<td>Define a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.</td>
<td>MVAR</td>
</tr>
<tr>
<td>Define a symbol that references a macro variable. ODS will convert the value of the variable to a number (stored as a double) before use. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.</td>
<td>NMVAR</td>
</tr>
<tr>
<td>Provide information about the table.</td>
<td>NOTES</td>
</tr>
<tr>
<td>Translate the specified numeric values to other values.</td>
<td>TRANSLATE-INTO</td>
</tr>
<tr>
<td>End a definition, or end the editing of a definition.</td>
<td>END</td>
</tr>
</tbody>
</table>

### Required Arguments

**table-path**

specifies where to store the table definition. 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.) PROC TEMPLATE writes the definition to the first template store in the current path that you can write to.

### Options

**STORE=libref.template-store**

specifies the template store in which to store the definition. If the template store does not exist, it is created.

**Restriction:** The STORE= option does not become part of the definition.
**Table Attributes**

This section lists all the attributes that you can use in a table definition. For all attributes that support a value of ON, the following forms are equivalent:

```
ATTRIBUTE-NAME
ATTRIBUTE-NAME=ON
```

For all of the attributes that support a value of variable, variable can be any variable that you declare in the table definition 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 the following table:

**Table 10.9  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 10.10  Table Attributes**

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Influence the layout of the table.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specify whether or not 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 LISTING, PRINTER family, and RTF</td>
</tr>
<tr>
<td></td>
<td>Specify whether or not to center each data panel independently if the entire table does not fit in one data panel.</td>
<td>CENTER LISTING, PRINTER family, RTF</td>
</tr>
<tr>
<td></td>
<td>Specify whether or not to force a new page before printing the table.</td>
<td>NEWPAGE All except OUTPUT</td>
</tr>
<tr>
<td></td>
<td>Specify the number of sets of columns to place on a page.</td>
<td>PANELS= LISTING and PRINTER family</td>
</tr>
<tr>
<td></td>
<td>Specify the number of blank characters to place between sets of columns when PANELS= is in effect.</td>
<td>PANEL_SPACE= LISTING</td>
</tr>
<tr>
<td>Task</td>
<td>Attribute</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Specify the number of lines that must be available on the page in</td>
<td>REQUIRED_SPACE=</td>
<td>LISTING and PRINTER family</td>
</tr>
<tr>
<td>order to print the body of the table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the number of lines to place between the previous output</td>
<td>TOP_SPACE=</td>
<td>LISTING and PRINTER family</td>
</tr>
<tr>
<td>object and the current one.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether or not to split a table that is too wide to fit in</td>
<td>WRAP</td>
<td>LISTING and PRINTER family</td>
</tr>
<tr>
<td>the space that is provided or to wrap each row of the table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify whether or not to add a double space after the last line of</td>
<td>WRAP_SPACE</td>
<td>LISTING and PRINTER family</td>
</tr>
<tr>
<td>a single row when the row is wrapped.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

_Influence the layout of rows and columns._

| Specify the maximum number of blank characters to place between      | COL_SPACE_MAX=       | LISTING                            |
| columns.                                                            |                      |                                     |
| Specify the minimum number of blank characters to place between      | COL_SPACE_MIN=       | LISTING                            |
| columns.                                                            |                      |                                     |
| Specify the name of the column whose value provides formatting       | CONTROL=             | All except OUTPUT                   |
| information about the space before each row of the definition.      |                      |                                     |
| Specify whether or not to double space between the rows of the      | DOUBLE_SPACE         | LISTING                            |
| table.                                                             |                      |                                     |
| Specify whether or not extra space is evenly divided among all       | EVEN                 | LISTING                            |
| columns of the table.                                               |                      |                                     |
| Specify whether or not to split a long stacked column across page    | SPLIT_STACK          | LISTING                            |
| boundaries.                                                        |                      |                                     |

_Influence the display of the values in header cells and data cells._

<p>| Specify whether or not to suppress blanking the value in a column    | CLASSLEVELS=         | LISTING and PRINTER family         |
| 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.                                                          |                      |                                     |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify which format to use if both a column definition and a data component specify a format.</td>
<td>DATA_FORMAT_OVERRIDE</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</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to order the columns by their order in the data component.</td>
<td>ORDER_DATA</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</td>
<td>All</td>
</tr>
<tr>
<td>Use the column name as the column header if neither the column definition nor the data component specifies a header.</td>
<td>USE_NAME</td>
<td>All</td>
</tr>
</tbody>
</table>

Influence the layout of headers and footers.

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<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=</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=</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not 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</td>
<td>LISTING</td>
</tr>
<tr>
<td>Specify whether or not to print table footers and any overlining of the table footers.</td>
<td>PRINT_FOOTERS</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether or not to print table headers and any underlining of the table headers.</td>
<td>PRINT_HEADERS</td>
<td>All except OUTPUT</td>
</tr>
<tr>
<td>Specify whether or not 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</td>
<td>LISTING</td>
</tr>
</tbody>
</table>

Influence the HTML output.
### Task Attribute Valid Destinations

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify whether or not to place the output object in a table of contents, if you create a table of contents.</td>
<td>CONTENTS</td>
<td>HTML, PDF, and PRINTER/PS PDFMARK</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=</td>
<td>HTML, PDF, PRINTER, PS PDFMARK</td>
</tr>
</tbody>
</table>

**Other table attributes.**

- Specify an alternate description for the table.\(^a\)  
  **ALT=** MARKUP
- Specify whether or not to print the current byline before the table.  
  **BYLINE=** All except OUTPUT
- Define the characters to use as the line-drawing characters in the table.  
  **FORMCHAR=** LISTING
- Specify a label for the table.  
  **LABEL=** All
- Specify a long description for the table.\(^a\)  
  **LONGDESC=** MARKUP
- Specify the table that the current definition inherits from.  
  **PARENT=** All
- Specify the style element to use for the table and any changes to the attributes.  
  **STYLE=** HTML, MARKUP family, PRINTER family, and RTF
- Specify the special data set type of a SAS data set.  
  **TYPE=** OUTPUT

---

\(^a\) SAS includes these accessibility and compatibility features that improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

**ALT= 'text'**

specifies an alternate description of the table.

**Requirement:** The text must be enclosed with quotation marks.

**ODS Destinations:** MARKUP

**Note:** SAS includes this accessibility and compatibility feature that improves the usability of SAS for users with disabilities. This feature is related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

**BALANCE <=ON | OFF | variable>**

specifies whether or not 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

**ODS Destinations:** LISTING, PRINTER family, and RTF
BYLINE <=ON | OFF | variable>
specifies whether or not to print the current byline before the table.

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.

ODS Destinations: All except OUTPUT

CENTER <=ON | OFF | variable>
specifies whether or not to center each data panel independently if the entire table does not fit in the space that is provided.

Default: ON

ODS Destinations: LISTING, PRINTER family, and RTF

CLASSLEVELS <=ON | OFF | variable>
specifies whether or not 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

ODS Destinations: All except OUTPUT

Featured in: Example 1 on page 342

COL_SPACE_MAX= positive-integer | variable
specifies the maximum number of blank characters to place between the columns.

Default: 4

ODS Destinations: LISTING

COL_SPACE_MIN= positive-integer | variable
specifies the minimum number of blank characters to place between the columns.

Default: 2

ODS Destinations: LISTING

CONTENTS <=ON | OFF | variable>
specifies whether or not to place the output object in a table of contents, if you create a table of contents.

Default: ON

ODS Destinations: HTML, PDF, and PRINTER/PS PDFMARK

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.

ODS Destinations: HTML, PDF, and PRINTER/PS PDFMARK

CONTROL=column-name | variable
specifies the name of the column whose values provide formatting information about the space before each row of the definition. The value of CONTROL= should be the name of a column of type character with a length equal to 1.
<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>'b' or 'B'</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

**ODS Destinations:** All except OUTPUT

**DATA_FORMAT_OVERRIDE=ON | OFF | variable**

specifies which format to use if both a column definition and a data component specify a format.

- **ON**
  - uses the format that the data component specifies.

- **OFF**
  - use the format that the column definition specifies.

**Default:** OFF

**ODS Destinations:** All

**DOUBLE_SPACE=ON | OFF | variable**

specifies whether or not to double space between the rows of the table.

**Default:** OFF

**ODS Destinations:** LISTING

**Featured in:** Example 1 on page 515 and Example 3 on page 528

**EVEN=ON | OFF | variable**

specifies whether or not extra space is evenly divided among all columns of the table.

**Default:** OFF

**ODS Destinations:** LISTING

**FOOTER_SPACE=0 | 1 | 2 | variable**

specifies the number of blank lines to place between the last row of data and the first row of the table footer.

**Default:** 1

**ODS Destinations:** LISTING

**FORMCHAR= 'string' | variable**

defines the characters to use as the line-drawing characters in the table. Currently, PROC TEMPLATE 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, you must specify both the first and second formatting characters. For example, the following option assigns the asterisk (*) to the first formatting character, the plus sign (+) to the second character, and does not alter the remaining characters:

`formchar='*+'`

**Default:** The SAS system option FORMCHAR= specifies the default formatting characters.
Tip: You can use any character in formatting-characters, including hexadecimal
characters. If you use hexadecimal characters, then you must put an x after the
closing quote. For example, the following option assigns the hexadecimal character
2D to the first formatting character, the hexadecimal character 7C to the second
character, and does not alter the remaining characters:

```
formchar='2D7C'x
```

**ODS Destinations:** LISTING

**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

**ODS Destinations:** LISTING

**JUSTIFY<=ON | OFF | variable>**
specifies whether to justify the format fields within the columns or to justify the
values within the columns without regard to the format fields.

**Default:** OFF

**Interaction:** JUSTIFY=ON can interfere with decimal alignment.

**Interaction:** If the column is numeric, then values are aligned to the right if you
specify JUSTIFY=OFF and JUST=C.

**Interaction:** All destinations except LISTING justify the values in columns as if
JUSTIFY=ON for JUST=R and JUST=L.

**Tip:** If you translate numeric data to character data, you might need to use
JUSTIFY= to align the data.

**Main discussion:** “How Are Values in Table Columns Justified?” on page 512

**ODS Destinations:** LISTING

**LABEL= ’text’ | variable**
specifies a label for the table.

**Default:** PROC TEMPLATE uses the first of the following that it finds:

- a label that the table definition provides
- a label that the data component provides
- the first spanning header in the table.

**ODS Destinations:** All

**LONGDESC= ’string’**
specifies the long description of the table.

**Requirement:** The text must be enclosed with quotation marks.

**ODS Destinations:** MARKUP

**Note:** SAS includes this accessibility and compatibility feature that improves the
usability of SAS for users with disabilities. This feature is related to accessibility
standards for electronic information technology adopted by the U.S. Government
under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. △

**NEWPAGE<=ON | OFF | variable>**
specifies whether or not 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.

**ODS Destinations:** All except OUTPUT
ORDER_DATA<=ON | OFF | variable>
specifies whether or not to order the columns by their order in the data component.

Default: OFF

When ORDER_DATA=OFF, the default order for columns is the order that they
are specified in the COLUMN statement. If you do not use a COLUMN statement,
the default order for columns is the order in which you define them in the
definition.

Tip: The Output destination always uses the order of the columns in the data
component when it creates an output data set.

Interaction: ORDER_DATA is most useful for ordering generic columns.

ODS Destinations: All except OUTPUT

OVERLINE<=ON | OFF | variable>
specifies whether or not 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. PROC TEMPLATE uses
the second formatting character to draw the line.

Default: OFF

Main discussion: See the discussion of FORMCHAR= on page 417.

See also: UNDERLINE= on page 421 (for tables), UNDERLINE= on page 387 (for
columns), and OVERLINE= on page 385 (for columns)

ODS Destinations: LISTING

Featured in: Example 1 on page 515

PANELS=positive-integer | variable
specifies the number of sets of columns to place on a page. If the width of all the
columns is less than half of the linesize, you can display the data in multiple sets of
columns so that rows that would otherwise appear on multiple pages appear on the
same page.

Tip: If the number of panels that is specified is larger than the number of panels
that can fit on the page, the definition creates as many panels as it can. Let the
table definition put your data in the maximum number of panels that can fit on
the page by specifying a large number of panels (for example, 99).

ODS Destinations: LISTING and PRINTER family

PANEL_SPACE=positive-integer | variable
specifies the number of blank characters to place between sets of columns when
PANELS= is in effect.

Default: 2

ODS Destinations: LISTING

PARENT=table-path
specifies the table that the current definition 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 definition
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 definition are used in the current definition unless the current
definition overrides them.

ODS Destinations: All

PRINT_FOOTERS<=ON | OFF | variable>
specifies whether or not to print table footers and any overlining of the table footers.

Default: ON
See also: OVERLINE= on page 419

ODS Destinations: All except OUTPUT

PRINT_HEADERS<=ON | OFF | variable>
specifies whether or not to print the table headers and any underlining of the table headers.
Default: ON
Interaction: When used in a table definition, PRINT_HEADERS affects only headers for the table, not the headers for individual columns. (See the discussion of the PRINT_HEADERS column attribute on page 386.)
See also: UNDERLINE= on page 421
ODS Destinations: All except OUTPUT

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
ODS Destinations: LISTING and PRINTER family

SPLIT_STACK<=ON | OFF | variable>
specifies whether or not to split a long stacked column across page boundaries.
Default: OFF
ODS Destinations: LISTING

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 definition that is registered with the Output Delivery System. SAS provides some style definitions. You can create your own style definitions with PROC TEMPLATE (see “DEFINE STYLE Statement” on page 288). 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 style definitions that SAS provides do not provide another style element that you would be likely to want to use instead of table. However, you may 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 can be either the name of a style element or a variable whose value is a style element.

See also: “Viewing the Contents of a Style Definition” on page 319
See also: “The Default Style Definition for HTML and Markup Languages” on page 320.

style-attribute-specification
describes the style attribute to change. Each style-attribute-specification has this general form:

    style-attribute-name=style-attribute-value

See also: “Style Attributes and Their Values” on page 292.

Default: table
Requirement: With the STYLE= option, you must specify either a style-attribute-specification or a style-element-name.
Tip: You can use braces ({ and }) instead of square brackets ([ and ]).

Tip: If you use the STYLE= attribute inside a quoted string, then you must 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 your quoted strings.

**ODS Destinations:** HTML, MARKUP family, PRINTER family, and RTF

**TOP_SPACE=** positive-integer | variable

specifies the number of lines to place between the previous output object and the current one.

Default: 1

**ODS Destinations:** LISTING and PRINTER family

**TYPE=** string | variable

specifies special type of SAS data set.

**Restriction:** PROC TEMPLATE does not verify the following:

- a SAS data set type that you specify is a valid data set type
- the structure of the data set that you create is appropriate for the type that you have assigned.

Tip: Most SAS data sets have no special type. However, certain SAS procedures, like the CORR procedure, can create a number of special SAS data sets. In addition, SAS/STAT software and SAS/EIS software support special data set types.

**ODS Destination:** OUTPUT

**UNDERLINE=<ON | OFF | variable>**

specifies whether or not 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). PROC TEMPLATE uses the second formatting character to draw the line.

Default: OFF

**Main discussion:** See the discussion of FORMCHAR= on page 417.

**See also:** OVERLINE= on page 419 (for tables), UNDERLINE= on page 387 (for columns), and OVERLINE= on page 385 (for columns)

**ODS Destinations:** LISTING

**Featured in:** Example 1 on page 515 and Example 3 on page 528

**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

**ODS Destinations:** All except OUTPUT

**USE_NAME=<ON | OFF | variable>**

uses the column name as the column header if neither the column definition nor the data component specifies a header.

Default: OFF
Tip: Use this attribute when column names are derived from a data set and the columns are generic.

ODS Destinations: All except OUTPUT

WRAP<=ON | OFF | variable>
specifies whether to split a wide table into multiple data panels, or to wrap each row of the table so that an entire row is printed before the next row starts.

Default: OFF

Interaction: When ODS wraps the rows of a table, it does not place multiple values in any column that contains an ID column.

See also: WRAP_SPACE= on page 422 and ID= on page 383

ODS Destinations: LISTING and PRINTER family

WRAP_SPACE<=ON | OFF | variable>
specifies whether or not to double space after the last line of a single row of the table when the row is wrapped onto more than one line.

Default: OFF

See also: WRAP= on page 422

ODS Destinations: LISTING, PRINTER family, and RTF

---

CELLSTYLE–AS Statement

Sets the style element of the cells in the table according to the values of the variables. Use this statement to set the presentation characteristics (such as foreground color, font face, flyover) of individual cells.

Featured in: Example 5 on page 539

CELLSTYLE expression-1 AS <style-element-name><[style-attribute-specification(s)]> <..., expression-n AS <style-element-name><[style-attribute-specification(s)]>>;

Required Arguments

equation

is an expression that is evaluated for each table cell that contains a variable. It can be any expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in SAS Language Reference: Dictionary. Use _VAL_ to represent the value of the current column. You may also reference symbols that you declared in a DYNAMIC, MVAR, or NVAR statement in the table definition.

If expression resolves to TRUE (a non-zero value), the style element that is specified is used for the current cell. If expression is FALSE (zero), the next expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

Restriction: You can not reference the values of other columns in expression.

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.
style-attribute-specification
describes a style attribute to set. Each style-attribute-specification has this general form:

\[ \text{style-attribute-name} = \text{style-attribute-value} \]

For information on the style attributes that you can set in a table definition, see “Style Attributes and Their Values” on page 292.

Options

style-element-name
is the name of a style element that is part of a style definition that is registered with the Output Delivery System. SAS provides some style definitions. You can create your own style definitions and style elements with PROC TEMPLATE. (See “DEFINE STYLE Statement” on page 288.)

The style elements that you would be most likely to use with the CELLSTYLE-AS statement are

- data
- datafixed
- dataempty
- dataemphasis
- dataemphasisfixed
- datastrong
- datastrongfixed.

The style element provides the basis for displaying the cell. Additional style attributes that you provide can modify the display.

COLUMN Statement

Declares a symbol as a column in the table and specifies the order of the columns

Featured in: Example 3 on page 528

\[ \text{COLUMN column(s);} \]

Required Arguments

column
is one or more columns. If the column is defined outside the current table definition, you must reference it by its path in the template store. Columns in the definition are laid out from left to right in the same order that you specify them in the COLUMN statement.

Default: If you do not use a COLUMN statement, ODS makes a column for each column definition (DEFINE COLUMN statement), and places the columns in the same order that the column definitions have in the table definition.
If you use a COLUMN statement but do not use a DEFINE COLUMN statement for any of the columns, ODS uses a default column definition that is based on the type of data in the column.

**Interaction:** If you specify the column attribute PRINT=OFF, then you turn off the value of a column if it is part of a stacked column. If all columns in a stacked column have PRINT=OFF set, then the entire column is removed from the table.

**Tip:** You can use a list of variable names, such as DAY1–DAY10, to specify multiple variables.

**Main discussion:** *Stacking Values for Two or More Variables*

To stack values for two or more variables in the same column, put parentheses around the variables that you want to stack. 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, the following COLUMN statement produces a definition in which

- 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.

```plaintext
column name (city state) (homephone workphone);
```

You can 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, the following COLUMN statement produces a report in which

- 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.

```plaintext
column name (city state) * (homephone workphone);
```

---

**DEFINE Statement**

*Creates a definition inside a table definition*

**Main discussion:** “DEFINE COLUMN Statement” on page 374, “DEFINE FOOTER Statement” on page 395, and “DEFINE HEADER Statement” on page 395

```plaintext
DEFINE definition-type definition-name</option(s)>;
                   statements-and-attributes
                  END;
```
Required Arguments

definition-type
specifies the type of definition to create, where definition-type is one of the following:

COLUMN
FOOTER
HEADER

The definition-type determines what other statements and what attributes can go in the definition. For details, see the documentation for the corresponding DEFINE statement.

definition-name
specifies the name of the new object.

Restriction: definition-name must be a single-level name.

Note: If you want to reference the definition that you are creating from another definition, you must create it outside the table definition.

Options

NOLIST
preserves the definition-type when inheriting it from another table definition.

Tip: If you specify an existing definition-name without using the NOLIST option, then the definition is overwritten.

DYNAMIC Statement

Defines a symbol that references a value that the data component supplies from the procedure or DATA step

Scope: You can use the DYNAMIC statement in the definition of a table, column, header, or footer. A dynamic variable that is defined in a definition is available to that definition and to all the definitions that it contains.

Featured in: Example 1 on page 342 and Example 2 on page 348

DYNAMIC variable-1 '<text-1>' <… variable-n '<text-n'>>;

Required Arguments

variable
Names a variable that the data component supplies. ODS resolves the value of the variable when it binds the definition and the data component.

Tip: Dynamic variables are most useful to the authors of SAS procedures and to DATA step programmers.
Options

text
is text that you can place in the definition to explain the dynamic variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

FOOTER Statement

Declares a symbol as a footer in the table and specifies the order of the footers

FOOTER footer-specification(s);

Required Arguments

footer-specification
is one or more footers. If the footer is defined outside the current table definition, you must reference it by its path in the template store. Footers in the definition are laid out from top to bottom in the same order that you specify them in the FOOTER statement. Each footer-specification can be

"string"
specifies the text to use for the footer. If you use a string, you do not need to use a DEFINE FOOTER statement. However, you cannot specify any footer attributes except for a split character. If the SPLIT= attribute is not in effect and if the first character of the footer that you specify is neither a blank character nor an alphanumeric character, PROC TEMPLATE treats it as the split character.

See also: SPLIT=.

footer-path
is the path of the footer definition to use. A footer-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 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 do not use a FOOTER statement, ODS makes a footer for each footer definition (DEFINE FOOTER statement), and places the footers in the same order that the footer definitions have in the table definition.

HEADER Statement

Declares a symbol as a header in the table and specifies the order of the headers

HEADER header-specification(s);
**Required Arguments**

**header-specification**

is one or more headers. If the header is defined outside the current table definition, you must reference it by its path in the template store. Headers in the definition are laid out from top to bottom in the same order that you specify them in the HEADER statement. Each header-specification can be

"string"

specifies the text to use for the header. If you use a string, you do not need to use a DEFINE HEADER statement. However, you cannot specify any header attributes except for a split character. If the SPLIT= header attribute is not in effect and if the first character of the header that you specify is neither a blank character nor an alphanumeric character, PROC TEMPLATE treats it as the split character.

See also: SPLIT=.

**header-path**

is the path of the header definition 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 do not use a HEADER statement, then ODS makes a header for each header definition (DEFINE HEADER statement), and places the headers in the same order that the header definitions have in the table definition.

**Featured in:** Example 3 on page 528

---

**MVAR Statement**

Defines a symbol that references a macro variable. ODS will use the value of the variable as a string. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.

Scope: You can use the MVAR statement in the definition of a table, column, header, or footer. A macro variable that is defined in a definition is available to that definition and to all the definitions that it contains.

**Featured in:** Example 3 on page 528 and Example 1 on page 342

MVAR variable-1 <"text-1"> <... variable-n <"text-n">>;
Required Arguments

**variable**
Names a macro variable to reference in the definition. ODS will use the value of the macro variable as a string. ODS does not resolve the value of the macro variable until it binds the definition and the data component.

**Tip:** You must declare macro variables this way in a definition. For example, to use the automatic macro variable SYSDATE9 in a definition, declare it in an MVAR statement and reference it as SYSDATE9, without an ampersand, in your PROC TEMPLATE step. If you use the ampersand, the macro variable resolves when the definition is compiled instead of when ODS binds the definition to the data component.

Options

**text**
is text that you can place in the definition to explain the macro variable's use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

---

**NMVAR Statement**

Defines a symbol that references a macro variable. ODS will convert the variable's value to a number (stored as a double) before using it. References to the macro variable are resolved when ODS binds the definition and the data component to produce an output object.

**Scope:** You can use the NMVAR statement in the definition of a table, column, header, or footer. A macro variable that is defined in a definition is available to that definition and to all the definitions that it contains.

**Featured in:** Example 5 on page 539

**NMVAR variable-1 <'text-1'> <… variable-n <'text-n'>>;;**

Required Arguments

**variable**
Names a macro variable to reference in the definition. ODS will convert the variable's value to a number (stored as a double) before using it. ODS does not resolve the macro variable until it binds the definition and the data component.

**Tip:** You must declare macro variables this way in a definition. 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 definition is compiled instead of when ODS binds the definition to the data component.
Options

text

is text that you can place in the definition to explain the macro variable’s use. Text of this type becomes part of the compiled definition, which you can view with the SOURCE statement, whereas SAS comments do not.

NOTES Statement

Provides information about the table

Tip: The NOTES statement becomes part of the compiled column definition, which you can view with the SOURCE statement, whereas SAS comments do not.

Featured in: Example 5 on page 539

NOTES 'text';

Required Arguments

text

provides information about the table.

TRANSLATE-INTO Statement

Translates the specified numeric values to other values

Restriction: The TRANSLATE-INTO statement in a table definition applies only to numeric variables. To translate the values of a character variable, use TRANSLATE-INTO in the definition of that column. (See “DEFINE COLUMN Statement” on page 374).

Featured in: Example 5 on page 539

TRANSLATE expression-1 INTO expression-2 <..., expression-n INTO expression-m>;

Required Arguments

expression-1

is an expression that is evaluated for each table cell that contains a numeric variable. It can be any numeric expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in SAS Language Reference: Dictionary. Use _VAL_ to represent the value of the current column. You may also reference symbols that you declared in a DYNAMIC, MVAR, or NVAR statement in the table definition.

If expression-1 resolves to TRUE (a non-zero value), the translation that is specified is used for the current cell. If expression-1 is FALSE (zero), the next
expression in the statement is evaluated. Thus, you can string multiple expressions together to format cells conditionally.

Restriction: You may not 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.

expression-2

is an expression that specifies the value to use in the cell in place of the variable’s actual value. It can be any expression that is valid in the WHERE statement (or the WHERE= data set option). For information on expressions that you can use in the WHERE statement, see “Statements” in SAS Language Reference: Dictionary. Use _VAL_ to represent the value of the current column. You may also reference symbols that you declared in a DYNAMIC, MVAR, or NVAR statement in the table definition.

Restriction: expression-2 must resolve to a character value, not a numeric value.

Restriction: You may not reference the values of other columns in expression-2.

Tip: When you translate a numeric value to a character value, the table definition 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. If you want the value to be right-justified, use the JUSTIFY=ON attribute.

See also: JUSTIFY= on page 418.

---

END Statement

Ends the table definition

END;

---

ODS Output Object Table Names

“ODS Table Names and the Base SAS Procedures that Produce Them” on page 430

“ODS Table Names and the SAS/STAT Procedures that Produce Them” on page 437

“ODS Table Names and the SAS/ETS Procedures that Produce Them” on page 490

---

ODS Table Names and the Base SAS Procedures that Produce Them

The following table lists the output object table names which Base SAS procedures produce. The table provides the name of each table, a description of what the table contains, and the option, if any, that creates the output object table. For more information about Base SAS procedures, see Base SAS Procedures Guide.
### Table 10.11  ODS Table Names Produced by the CALENDAR Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>Calendar</td>
</tr>
</tbody>
</table>

### Table 10.12  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 10.13  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 10.14  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 (Comparison results for observations)</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>
</tbody>
</table>
## Table Name | Description | Option
---|---|---
**CompareSummary** | Summary report of observations, values, and variables of unequal values | Omit NOSUMMARY option or unless the variables are identical

**CompareVariables** | List of differences in variable types or attributes between the base data set and the compare data set | 

### ODS Tables Created by the ID Statement

**CompareDetails** | List of notes and warnings concerning duplicate ID variable values, if duplicate ID variable values exist in either the data set | 

## Table 10.15 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>Covariance table</td>
<td>COV</td>
</tr>
<tr>
<td></td>
<td>row/column variance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DF (missing values)</td>
<td></td>
</tr>
<tr>
<td>CronbachAlpha</td>
<td>Coefficient alpha</td>
<td>ALPHA</td>
</tr>
<tr>
<td>CronbachAlphaDel</td>
<td>Coefficient alpha with deleted variables</td>
<td>ALPHA</td>
</tr>
<tr>
<td>Ccscp</td>
<td>Corrected sums of squares and crossproducts</td>
<td>CSSCP</td>
</tr>
<tr>
<td></td>
<td>Row/column variable corrected sums of squares (missing values)</td>
<td></td>
</tr>
<tr>
<td>HoeffdingCorr</td>
<td>Hoeffding's D statistics</td>
<td>HOEFFDING</td>
</tr>
<tr>
<td></td>
<td>p-value (NOPROB is not specified)</td>
<td></td>
</tr>
<tr>
<td>KendallCorr</td>
<td>Kendall tau-b coefficients</td>
<td>Pearson or omit NOCORR option</td>
</tr>
<tr>
<td></td>
<td>p-value (NOPROB is not specified)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of observations (missing values)</td>
<td></td>
</tr>
<tr>
<td>SimpleStats</td>
<td>Simple descriptive statistics</td>
<td>Omit NOSIMPLE option</td>
</tr>
<tr>
<td>SpearmanCorr</td>
<td>Spearman descriptive statistics</td>
<td>SPEARMAN</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Sscp</td>
<td>Sums of squares and crossproducts</td>
<td>SSCP</td>
</tr>
<tr>
<td></td>
<td>Row/column variable sums of squares (missing values)</td>
<td></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>PartialCcsscp</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 Kendall tau-b coefficients, p-values (NOPROB option is not specified)</td>
<td></td>
</tr>
<tr>
<td>PartialSpearmanCorr</td>
<td>Partial Spearman correlations, p-values (NOPROB option is not specified)</td>
<td>SPEARMAN</td>
</tr>
</tbody>
</table>

**Table 10.16** 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 10.17** 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 option</td>
</tr>
<tr>
<td>EngineHost</td>
<td>Engine and operating environment information</td>
<td>Omit SHORT option</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------</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 option</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>
<tr>
<td>SortedbyShort</td>
<td>Concise sort information</td>
<td>SHORT option and data set is sorted</td>
</tr>
<tr>
<td>Variables</td>
<td>List of variables in alphabetical order</td>
<td>Omit SHORT option</td>
</tr>
<tr>
<td>VariablesShort</td>
<td>Concise listing of variables in alphabetical order</td>
<td>SHORT</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 10.18 ODS Table Names Produced by the FREQ Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinomialProp</td>
<td>Binomial proportion</td>
<td>BINOMIAL (one-way tables)</td>
</tr>
<tr>
<td>BinomialPropTest</td>
<td>Binomial proportion test</td>
<td>BINOMIAL (one-way tables)</td>
</tr>
<tr>
<td>BreslowDayTest</td>
<td>Breslow-day test</td>
<td>CMH (hx2x2 tables)</td>
</tr>
<tr>
<td>CMH</td>
<td>Cochran-Mantel-Haenszel statistics</td>
<td>CMH</td>
</tr>
<tr>
<td>ChiSq</td>
<td>Chi-Square tests and measures</td>
<td>CHISQ</td>
</tr>
<tr>
<td>CochransQ</td>
<td>Cochran’s Q</td>
<td>AGREE (hx2x2 tables)</td>
</tr>
<tr>
<td>ColScores</td>
<td>Column scores</td>
<td>SCOROUT</td>
</tr>
<tr>
<td>CommonRelRisks</td>
<td>Common relative risks</td>
<td>CMH (hx2x2 tables)</td>
</tr>
<tr>
<td>Crosslist</td>
<td>Cross lists</td>
<td></td>
</tr>
<tr>
<td>CrossTabFreqs</td>
<td>Crosstabulation table</td>
<td>(n-way table request, n&gt;1)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>EqualKappaTest</td>
<td>Test for equal simple kappas</td>
<td>AGREE (hx2x2 tables)</td>
</tr>
<tr>
<td>EqualKappaTests</td>
<td>Test for equal kappas</td>
<td>AGREE (hxrxr tables, r&gt;2)</td>
</tr>
<tr>
<td>FishersExact</td>
<td>Fisher's exact test</td>
<td>FISHER or EXACT CHISQ (2x2 tables)</td>
</tr>
<tr>
<td>JTTest</td>
<td>Jonckheere-Terpstra test</td>
<td>JT</td>
</tr>
<tr>
<td>KappaStatistics</td>
<td>Kappa statistics</td>
<td>AGREE (rxr tables, r&gt;2, and no TEST or EXACT requests for kappas)</td>
</tr>
<tr>
<td>KappaWeight</td>
<td>Kappa weights</td>
<td>AGREE and PRINTKWT</td>
</tr>
<tr>
<td>List</td>
<td>List frequencies</td>
<td>LIST</td>
</tr>
<tr>
<td>McNemarsTest</td>
<td>McNemar's test</td>
<td>AGREE (2x2 tables)</td>
</tr>
<tr>
<td>Measures</td>
<td>Measures of association</td>
<td>MEASURES</td>
</tr>
<tr>
<td>OneWayChiSq</td>
<td>One-way Chi-Square goodness-of-fitnes test</td>
<td>CHISQ (one-way tables)</td>
</tr>
<tr>
<td>OneWayFreqs</td>
<td>One-way frequencies</td>
<td>(one-way table request)</td>
</tr>
<tr>
<td>OverallKappa</td>
<td>Overall simple kappa coefficient</td>
<td>AGREE (hx2x2 tables)</td>
</tr>
<tr>
<td>Overallkappas</td>
<td>Overall kappa coefficients</td>
<td>AGREE (hxrxr tables, r&gt;2)</td>
</tr>
<tr>
<td>RelativeRisks</td>
<td>Relative risk estimates</td>
<td>RELRISK or MEASURES (2x2 tables)</td>
</tr>
<tr>
<td>RiskDiffCol1</td>
<td>Column 1 risk estimates</td>
<td>RISKDIFF (2x2 tables)</td>
</tr>
<tr>
<td>RiskDiffCol2</td>
<td>Column 2 risk estimates</td>
<td>RISKDIFF (2x2 tables)</td>
</tr>
<tr>
<td>RowScores</td>
<td>Row scores</td>
<td>SCOROUT</td>
</tr>
<tr>
<td>SimpleKappaTest</td>
<td>Simple kappa tests</td>
<td>AGREE (2x2 tables), AGREE (rxr tables, r&gt;2)</td>
</tr>
<tr>
<td>SymmetryTest</td>
<td>Test of symmetry</td>
<td>AGREE</td>
</tr>
<tr>
<td>TrendTest</td>
<td>Cochran-Armitage test for trend</td>
<td>TREND</td>
</tr>
<tr>
<td>WeightKappa</td>
<td>Weighted kappa coefficient</td>
<td>AGREE (rxr tables, r&gt;2)</td>
</tr>
</tbody>
</table>

Table 10.19 ODS Table Names Produced by the MEANS and SUMMARY Procedures

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Summary of descriptive statistics for variables across all observations and within groups of observations</td>
</tr>
</tbody>
</table>
### Table 10.20  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 10.21  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 10.22  ODS Table Names Produced by the SQL Procedure

<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 10.23  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 10.24  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 10.25  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></td>
</tr>
<tr>
<td>ExtremeObs</td>
<td>Extreme observations</td>
<td></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></td>
</tr>
<tr>
<td>Modes</td>
<td>Modes</td>
<td>MODES</td>
</tr>
<tr>
<td>Moments</td>
<td>Sample moments</td>
<td></td>
</tr>
<tr>
<td>Plots</td>
<td>Line printer plots</td>
<td>PLOTS</td>
</tr>
<tr>
<td>Quantiles</td>
<td>Quantiles</td>
<td></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></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 Table Names and the SAS/STAT Procedures that Produce Them**

The following table lists the output object table names which SAS/STAT procedures produce. You must license SAS/STAT software in order to produce these output objects. The table provides the name of each table, a description of what the table contains, and
the option, if any, that creates the output object table. For information about SAS/STAT procedures, see

Table 10.26  ODS Table Names Produced by the ACECLUS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>DataOptionInfo</td>
<td>Data and option information</td>
<td></td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>Eigenvalues of (\text{Inv(ACE) + COV-ACE})</td>
<td></td>
</tr>
<tr>
<td>Eigenvectors</td>
<td>Eigenvectors (raw canonical coefficients)</td>
<td></td>
</tr>
<tr>
<td>InitWithin</td>
<td>Initial within-cluster covariances estimate</td>
<td>INITIAL=INPUT</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td></td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td></td>
</tr>
<tr>
<td>StdCanCoef</td>
<td>Standardized canonical coefficients</td>
<td></td>
</tr>
<tr>
<td>Threshold</td>
<td>Threshold value</td>
<td>PROPORTION=</td>
</tr>
<tr>
<td>TotSampleCov</td>
<td>Total sample covariances</td>
<td></td>
</tr>
<tr>
<td>Within</td>
<td>Approximate covariance estimate within clusters</td>
<td></td>
</tr>
</tbody>
</table>

Table 10.27  ODS Table Names Produced by the ANOVA Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DependentInfo</td>
<td>Simultaneously analyzed dependent variables</td>
<td>default when there are multiple dependent variables with different patterns of missing values</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>R-Square, C.V, root MSE, and dependent mean</td>
<td></td>
</tr>
<tr>
<td>ModelANOVA</td>
<td>ANOVA for model terms</td>
<td></td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations</td>
<td></td>
</tr>
<tr>
<td>OverallANOVA</td>
<td>Overall ANOVA</td>
<td></td>
</tr>
</tbody>
</table>

ODS Tables Created by the CLASS Statement
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassLevels</td>
<td>Classification variable levels</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MANOVA Statement**

- **MANOVATransform**
  - Description: Multivariate transformation matrix
  - Option: \( M= \)
- **MultStat**
  - Description: Multivariate tests
- **Tests**
  - Description: Summary ANOVA for specified MANOVA \( H= \) effects
  - Option: \( H=\text{SUMMARY} \)

**ODS Tables Created by the MANOVA or REPEATED Statements**

- **CanAnalysis**
  - Description: Canonical analysis
  - Option: \( \text{CANONICAL} \)
- **CanCoeff**
  - Description: Canonical coefficients
  - Option: \( \text{CANONICAL} \)
- **CanStructure**
  - Description: Canonical structure
  - Option: \( \text{CANONICAL} \)
- **CharStruct**
  - Description: Characteristic roots and vectors
  - Option: MANOVA (not \( \text{CANONICAL} \)); REPEATED \( \text{PRINTR} \)
- **ErrorSSCP**
  - Description: Error SSCP matrix
  - Option: \( \text{PRINT} \)
- **HypothesisSSCP**
  - Description: Hypothesis SSCP matrix
  - Option: \( \text{PRINT} \); MANOVA \( M= \)
- **PartialCorr**
  - Description: Partial correlation matrix
  - Option: \( \text{PRINT} \); REPEATED \( (\text{CONTRAST}, \text{HELMERT}, \text{MEAN}, \text{POLYNOMIAL}, \text{or PROFILE}) \)

**ODS Tables Created by the MEANS Statement**

- **Bartlett**
  - Description: Bartlett’s homogeneity of variance test
  - Option: \( \text{HOVTEST}=\text{BARTLETT} \)
- **CLDiffs**
  - Description: Multiple comparisons of pairwise differences
  - Option: \( \text{CLDIFF or DUNNETT or} \) (Unequal cells and not \( \text{LINES} \))
- **CLDiffsInfo**
  - Description: Information for multiple comparisons of pairwise differences
  - Option: \( \text{CLDIFF or DUNNETT or} \) (Unequal cells and not \( \text{LINES} \))
- **CLMeans**
  - Description: Multiple comparisons of means with confidence/comparison interval
  - Option: \( \text{CLM with (BON, GABRIEL, SCHEEFE, SIDAL, SMM, T, or LSD)} \)
- **CLMeansInfo**
  - Description: Information for multiple comparisons of means with confidence/comparison interval
  - Option: \( \text{CLM} \)
- **HOVFTest**
  - Description: Homogeneity of variance ANOVA
  - Option: \( \text{HOVTEST} \)
- **MCLines**
  - Description: Multiple comparisons LINES output
  - Option: \( \text{LINES, (DUNCAN or WALLER or SNK or REGWQ) \ and not (CLDIFF or CLM)}, \) \( \text{or (equal cells and not CLDIFF)} \)
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLinesInfo</td>
<td>Information for multiple comparison LINES output</td>
<td>LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (equal cells and not CLDIFF)</td>
</tr>
<tr>
<td>MCLinesRange</td>
<td>Ranges for multiple range MC tests</td>
<td>LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (equal cells and not CLDIFF)</td>
</tr>
<tr>
<td>Means</td>
<td>Group means</td>
<td>WELCH</td>
</tr>
<tr>
<td>Welch</td>
<td>Welch's ANOVA</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the REPEATED Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epsilons</td>
<td>Greenhouse-Geisser and Huynh-Feldt epsilons</td>
<td></td>
</tr>
<tr>
<td>RepTransform</td>
<td>Repeated transformation matrix</td>
<td>CONTRAST, HELMERT, MEAN, POLYNOMIAL, or PROFILE</td>
</tr>
<tr>
<td>RepeatedLevelInfo</td>
<td>Correspondence between dependents and repeated measures levels</td>
<td></td>
</tr>
<tr>
<td>Sphericity</td>
<td>Sphericity tests</td>
<td>PRINTE</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the TEST Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AltErrTests</td>
<td>ANOVA tests with error other than MSE</td>
<td>E=</td>
</tr>
</tbody>
</table>

**Table 10.28** ODS Table Names Produced by the CALIS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS Tables Created by the COSAN, FACTOR, LINEQS, and RAM Models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddParms</td>
<td>Additional parameters in the PARAMETERS statement</td>
<td>PINITIAL or default</td>
</tr>
<tr>
<td>AsymStdRes</td>
<td>Asymptotically standardized residual matrix</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>AveAsymStdRes</td>
<td>Average absolute asymptotically standardized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>AveNormRes</td>
<td>Average absolute normalized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>AveRawRes</td>
<td>Average absolute raw residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>AveVarStdRes</td>
<td>Average absolute variance standardized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>ContKurtosis</td>
<td>Contributions to kurtosis</td>
<td>KURTOSIS or PRINT</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td>PSHORT</td>
</tr>
<tr>
<td>CorrParm</td>
<td>Correlations among parameter estimates</td>
<td>PCOVES and default</td>
</tr>
<tr>
<td>CovMat</td>
<td>Assorted cov matrices</td>
<td>PCOVES and default</td>
</tr>
<tr>
<td>DependParms</td>
<td>Dependent parameters (if specified by program statements)</td>
<td>PRIVEC and default</td>
</tr>
<tr>
<td>DistAsymStdRes</td>
<td>Distribution of asymptotically standardized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>DistNormRes</td>
<td>Distribution of normalized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>DistVarStdRes</td>
<td>Distribution of variance standardized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>Estimates</td>
<td>Vector of estimates</td>
<td>PRIVEC</td>
</tr>
<tr>
<td>Fit</td>
<td>Fit statistics</td>
<td>PSUMMARY</td>
</tr>
<tr>
<td>GenModInfo</td>
<td>General modeling information</td>
<td>PSIMPLE or default</td>
</tr>
<tr>
<td>Gradient</td>
<td>First partial derivatives (Gradient)</td>
<td>PRIVEC and default</td>
</tr>
<tr>
<td>InCorr</td>
<td>Input correlation matrix</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>InCorrDet</td>
<td>Determinant of the input correlation matrix</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>InCov</td>
<td>Input covariance matrix</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>InCovDet</td>
<td>Determinant of the input covariance matrix</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>Information</td>
<td>Information matrix</td>
<td>PCOVES and default</td>
</tr>
<tr>
<td>InitEstimates</td>
<td>Initial vector of parameter estimates</td>
<td>PINITIAL or default</td>
</tr>
<tr>
<td>InSymmetric</td>
<td>Input symmetric matrix (SYMATRIX data type)</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>IterHist</td>
<td>Iteration history</td>
<td>PSHORT</td>
</tr>
<tr>
<td>IterStart</td>
<td>Iteration start</td>
<td>PSHORT</td>
</tr>
<tr>
<td>IterStop</td>
<td>Iteration stop</td>
<td>PSHORT</td>
</tr>
<tr>
<td>Jacobian</td>
<td>Jacobi column pattern</td>
<td>PJACPAT</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>Kurtosis, with raw data input</td>
<td>KURTOSIS or PRINT</td>
</tr>
<tr>
<td>LagrangeBoundary</td>
<td>Lagrange, releasing active boundary constraints</td>
<td>MODIFICATION or PALL</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>LagrangeEquality</td>
<td>Lagrange, releasing equality constraints</td>
<td>MODIFICATION or PALL</td>
</tr>
<tr>
<td>ModelStatement</td>
<td>Model summary</td>
<td>PSHORT</td>
</tr>
<tr>
<td>ModIndices</td>
<td>Lagrange multiplier and Wald test statistics</td>
<td>MODIFICATION or PALL</td>
</tr>
<tr>
<td>NormRes</td>
<td>Normalized residual matrix</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>PredetElements</td>
<td>Predetermined elements</td>
<td>PREDET or PALL</td>
</tr>
<tr>
<td>PredModel</td>
<td>Predicted model matrix</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>PredModelDet</td>
<td>Predicted model determinant</td>
<td>PCORR or PALL</td>
</tr>
<tr>
<td>ProblemDescription</td>
<td>Problem Description</td>
<td>PSHORT</td>
</tr>
<tr>
<td>RankAsymStdRes</td>
<td>Ranking of the largest asymptotically standardized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>RankLagrange</td>
<td>Ranking of the largest Lagrange indices</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>RankNormRes</td>
<td>Ranking of the largest normalized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>RankRawRes</td>
<td>Ranking of the largest raw residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>RankVarStdRes</td>
<td>Ranking of the largest variance standardized residuals</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>RawRes</td>
<td>Raw residual matrix</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics, with raw data input</td>
<td>SIMPLE or default</td>
</tr>
<tr>
<td>StdErrs</td>
<td>Vector of standard errors</td>
<td>PRIVEC and default</td>
</tr>
<tr>
<td>SumSqDif</td>
<td>Sum of squared differences of predetermined elements</td>
<td>PREDET or PALL</td>
</tr>
<tr>
<td>tValues</td>
<td>Vector of t values</td>
<td>PRIVEC and default</td>
</tr>
<tr>
<td>VarStdRes</td>
<td>Variance of standardized residual matrix</td>
<td>RESIDUAL= or PRINT</td>
</tr>
<tr>
<td>WaldTest</td>
<td>Wald test</td>
<td>MODIFICATION or PALL</td>
</tr>
<tr>
<td>Weights</td>
<td>Weight matrix</td>
<td>PWEIGHT or PALL</td>
</tr>
<tr>
<td>WeightsDet</td>
<td>Determinant of the weight matrix</td>
<td>PWEIGHT or PALL</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the FACTOR, LINEQS, and RAM Models**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determination</td>
<td>Coefficients of determination</td>
<td>PDETERM and default</td>
</tr>
<tr>
<td>SqMultCorr</td>
<td>Squared multiple correlations</td>
<td>PESTIM or PSHORT</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the COSAN and FACTOR Models**
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>EstParms</td>
<td>Estimated parameter matrix</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>InitParms</td>
<td>Initial matrix of parameter estimates</td>
<td>PINITIAL or default</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the LINEQS and RAM Models**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indirect Effects</td>
<td>Indirect effects</td>
<td>TOTEFF or PRINT</td>
</tr>
<tr>
<td>InitParms</td>
<td>Initial matrix of parameter estimates</td>
<td>PRIMAT and default</td>
</tr>
<tr>
<td>LatentScoreCoef</td>
<td>Latent variable regression score coefficients</td>
<td>PLATCOV or PRINT</td>
</tr>
<tr>
<td>PredMomentLatent</td>
<td>Predicted latent variable moments</td>
<td>PLATCOV or PRINT</td>
</tr>
<tr>
<td>PredMomentManLat</td>
<td>Predicted manifest and latent variable moments</td>
<td>PLATCOV or PRINT</td>
</tr>
<tr>
<td>SetCovExog</td>
<td>Set covariance parameters for manifest exogenous variables</td>
<td>PINITIAL or default</td>
</tr>
<tr>
<td>Stability</td>
<td>Stability of reciprocal causation</td>
<td>PDETERM and default</td>
</tr>
<tr>
<td>StructEq</td>
<td>Variables in the structural equations</td>
<td>PDETERM and default</td>
</tr>
<tr>
<td>TotalEffects</td>
<td>Total effects</td>
<td>TOTEFF or PRINT</td>
</tr>
<tr>
<td>VarSelection</td>
<td>Manifest variables, if not all are used, selected for modeling</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the FACTOR Model**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FactCorrExog</td>
<td>Correlations among factors</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>FactScoreCoef</td>
<td>Factor score regression coefficients</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>RotatedLoadings</td>
<td>Rotated loadings, with ROTATE= option in FACTOR statement</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>Rotation</td>
<td>Rotation matrix, with ROTATE= option in FACTOR statement</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>StdLoadings</td>
<td>Standardized factor loadings</td>
<td>PESTIM or PSHORT</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the LINEQS Model**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CorrExog</td>
<td>Correlations among exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>EndogenousVar</td>
<td>Endogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>EstCovExog</td>
<td>Estimated covariances among exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>EstLatentEq</td>
<td>Estimated latent variable equations</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>EstManifestEq</td>
<td>Estimated manifest variable equations</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>EstVarExog</td>
<td>Estimated variances of exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>ExogenousVar</td>
<td>List of exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>InCovExog</td>
<td>Input covariances among exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>InLatentEq</td>
<td>Input latent variable equations</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>InManifestEq</td>
<td>Input manifest variable equations</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>InVarExog</td>
<td>Input variances of exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>StdLatentEq</td>
<td>Standardized latent variable equations</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>StdManifestEq</td>
<td>Standardized manifest variable equations</td>
<td>PESTIM or PSHORT</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the RAM Model**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitRAMEstimates</td>
<td>Initial RAM estimates</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>RAMCorrExog</td>
<td>Correlations among exogenous variables</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>RAMEstimates</td>
<td>RAM final estimates</td>
<td>PESTIM or PSHORT</td>
</tr>
<tr>
<td>RAMStdEstimates</td>
<td>Standardized estimates</td>
<td>PESTIM or PSHORT</td>
</tr>
</tbody>
</table>

**Table 10.29** ODS Table Names Produced by the CANCORR Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultStat</td>
<td>Multivariate statistics</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by PROC CANCORR**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvgRSquare</td>
<td>Average R-Squares (weighted and unweighted)</td>
<td>VDEP (or WDEP) or SMC (or ALL)</td>
</tr>
<tr>
<td>CanCorr</td>
<td>Canonical correlations</td>
<td></td>
</tr>
<tr>
<td>CanStructureVCan</td>
<td>Correlations between the VAR canonical variables and the VAR and WITH variables</td>
<td>default (unless SHORT)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>CanStructureWCan</td>
<td>Correlations between the WITH canonical variables and the WITH and VAR variables</td>
<td>default (unless SHORT)</td>
</tr>
<tr>
<td>ConfidenceLimits</td>
<td>95% confidence limits for the regression coefficients</td>
<td>VDEP (or WDEP) or CLB (or ALL)</td>
</tr>
<tr>
<td>Corr</td>
<td>Correlations among the original variables</td>
<td>CORR (or ALL)</td>
</tr>
<tr>
<td>CorrRegCoeffEst</td>
<td>Correlations among the regression coefficient estimates</td>
<td>VDEP (or WDEP) or CORRB (or ALL)</td>
</tr>
<tr>
<td>NObsNVar</td>
<td>Number of observations and variables</td>
<td>SIMPLE (or ALL)</td>
</tr>
<tr>
<td>ParCorr</td>
<td>Partial correlations</td>
<td>VDEP (or WDEP) or PCORR (or ALL)</td>
</tr>
<tr>
<td>ProbtRegCoeff</td>
<td>Prob &gt;</td>
<td>t</td>
</tr>
<tr>
<td>RawCanCoeffV</td>
<td>Raw canonical coefficients for the VAR variables</td>
<td>default (unless SHORT)</td>
</tr>
<tr>
<td>RawCanCoeffW</td>
<td>Raw canonical coefficients for the WITH variables</td>
<td>default (unless SHORT)</td>
</tr>
<tr>
<td>RawRegCoeff</td>
<td>Raw regression coefficients</td>
<td>VDEP (or WDEP) or B (or ALL)</td>
</tr>
<tr>
<td>Redundancy</td>
<td>Canonical redundancy analysis</td>
<td>REDUNDANCY (or ALL)</td>
</tr>
<tr>
<td>Regression</td>
<td>Squared multiple correlations and F tests</td>
<td>VDEP (or WDEP) or SMC (or ALL)</td>
</tr>
<tr>
<td>SemiParCorr</td>
<td>Semi-partial correlations</td>
<td>VDEP (or WDEP) or SPCORR (or ALL)</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>SIMPLE (or ALL)</td>
</tr>
<tr>
<td>SqMultCorr</td>
<td>Canonical redundancy analysis: squared multiple correlations</td>
<td>REDUNDANCY (or ALL)</td>
</tr>
<tr>
<td>SqParCorr</td>
<td>Squared partial correlations</td>
<td>VDEP (or WDEP) or SQPCORR (or ALL)</td>
</tr>
<tr>
<td>SqSemiParCorr</td>
<td>Squared semi-partial correlations</td>
<td>VDEP (or WDEP) or SQSPCORR (or ALL)</td>
</tr>
<tr>
<td>StdCanCoeffV</td>
<td>Standardized canonical coefficients for the VAR variables</td>
<td>default (unless SHORT)</td>
</tr>
<tr>
<td>StdCanCoeffW</td>
<td>Standardized canonical coefficients for the WITH variables</td>
<td>default (unless SHORT)</td>
</tr>
<tr>
<td>StdErrRawRegCoeff</td>
<td>Standard errors of the raw regression coefficients</td>
<td>VDEP (or WDEP) or SEB (or ALL)</td>
</tr>
<tr>
<td>StdRegCoeff</td>
<td>Standardized regression coefficients</td>
<td>VDEP (or WDEP) or STB (or ALL)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>tValueRegCoef</td>
<td>t values for the regression coefficients</td>
<td>VDEP (or WDEP) or T (or ALL)</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>CorrOnPartial</td>
<td>Partial correlations</td>
<td>CORR (or ALL)</td>
</tr>
<tr>
<td>RSquareRMSEOnPartial</td>
<td>R-Squares and RMSEs on PARTIAL</td>
<td>CORR (or ALL)</td>
</tr>
<tr>
<td>StdRegCoefOnPartial</td>
<td>Standardized regression coefficients on PARTIAL</td>
<td>CORR (or ALL)</td>
</tr>
</tbody>
</table>

**Table 10.30 ODS Table Names Produced by the CANDISC Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Univariate statistics</td>
<td>ANOVA</td>
</tr>
<tr>
<td>AveRSquare</td>
<td>Average R-Square</td>
<td>ANOVA</td>
</tr>
<tr>
<td>BCorr</td>
<td>Between-class correlations</td>
<td>BCORR</td>
</tr>
<tr>
<td>BCov</td>
<td>Between-class covariances</td>
<td>BCOV</td>
</tr>
<tr>
<td>BSSCP</td>
<td>Between-class SSCP matrix</td>
<td>BSSCP</td>
</tr>
<tr>
<td>BStruct</td>
<td>Between canonical structure</td>
<td></td>
</tr>
<tr>
<td>CanCorr</td>
<td>Canonical correlations</td>
<td></td>
</tr>
<tr>
<td>CanonicalMeans</td>
<td>Class means on canonical variables</td>
<td></td>
</tr>
<tr>
<td>Counts</td>
<td>Number of observations, variables, classes, DF</td>
<td></td>
</tr>
<tr>
<td>CovDF</td>
<td>DF for covariance matrices, not printed</td>
<td>any *COV option</td>
</tr>
<tr>
<td>Dist</td>
<td>Squared distances</td>
<td>MAHALANOBIS</td>
</tr>
<tr>
<td>DistFValues</td>
<td>F statistics based on squared distances</td>
<td>MAHALANOBIS</td>
</tr>
<tr>
<td>DistProb</td>
<td>Probabilities for F statistics from squared distances</td>
<td>MAHALANOBIS</td>
</tr>
<tr>
<td>Levels</td>
<td>Class level information</td>
<td></td>
</tr>
<tr>
<td>MultStat</td>
<td>MANOVA</td>
<td></td>
</tr>
<tr>
<td>PCoeef</td>
<td>Pooled standard canonical coefficients</td>
<td></td>
</tr>
<tr>
<td>PCorr</td>
<td>Pooled within-class correlations</td>
<td>PCORR</td>
</tr>
<tr>
<td>PCov</td>
<td>Pooled within-class covariances</td>
<td>PCOV</td>
</tr>
</tbody>
</table>
### Table 10.31 ODS Table Names Produced by the CATMOD Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Analysis of variance</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td>ML</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation matrix of the</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance matrix of the</td>
<td>COVB</td>
</tr>
<tr>
<td>Estimates</td>
<td>Analysis of estimates</td>
<td>default, unless NOPARM</td>
</tr>
<tr>
<td>MaxLikelihood</td>
<td>Maximum likelihood analysis</td>
<td>ML</td>
</tr>
<tr>
<td>OneWayFreqs</td>
<td>One-way frequencies</td>
<td>ONEWAY</td>
</tr>
<tr>
<td>PopProfiles</td>
<td>Population profiles</td>
<td>default, unless NOPROFILE</td>
</tr>
<tr>
<td>PredictedFreqs</td>
<td>Predicted frequencies</td>
<td>PRED=FREQ</td>
</tr>
<tr>
<td>PredictedProbs</td>
<td>Predicted probabilities</td>
<td>PREDICT or PRED=PROB</td>
</tr>
<tr>
<td>PredictedValues</td>
<td>Predicted values</td>
<td>PREDICT or PRED=</td>
</tr>
</tbody>
</table>
### Table Name | Description | Option
--- | --- | ---
ResponseCov | Response functions, covariance matrix | COV
ResponseDesign | Response functions, design matrix | WLS, unless NODESIGN
ResponseFreqs | Response frequencies | FREQ
ResponseProbs | Response probabilities | PROB
ResponseProfiles | Response profiles | default, unless NOPROFILE
XPX | $X^*\text{Inv}(S)^*X$ matrix | XPX, for WLS

**ODS Tables Created by the CONTRAST Statement**
- Contrasts
- ContrastEstimates: Analysis of contrasts

**ODS Tables Created by the PROC Statement**
- DataSummary

**ODS Tables Created by the MODEL and LOGLIN Statements**
- ResponseMatrix: _RESPONSE_ matrix

### Table 10.32  ODS Table Names Produced by the CLUSTER Procedure

| Table Name | Description | Option |
--- | --- | ---
ClusterHistory | Observations or clusters joined, frequencies and other cluster statistics | 
SimpleStatistics | Simple statistics, before or after trimming | SIMPLE
EigenvalueTable | Eigenvalues of the CORR or COV matrix |
Table 10.33 ODS Table Names Produced by the CORRESP Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdjInGreenacre</td>
<td>Greenacre inertia adjustment</td>
<td>GREENACRE</td>
</tr>
<tr>
<td>AdjInBenzecri</td>
<td>Benzecri inertia adjustment</td>
<td>BENZECRI</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary table</td>
<td></td>
</tr>
<tr>
<td>BinaryPct</td>
<td>Binary table percents</td>
<td>OBSERVED or BINARY</td>
</tr>
<tr>
<td>Burt</td>
<td>Burt table</td>
<td></td>
</tr>
<tr>
<td>BurtPct</td>
<td>Burt table percents</td>
<td>OBSERVED or MCA</td>
</tr>
<tr>
<td>CellChiSq</td>
<td>Contributions to Chi Square</td>
<td>CELLCHI2</td>
</tr>
<tr>
<td>CellChiSqPct</td>
<td>Contributions, percents</td>
<td>CELLCHI2</td>
</tr>
<tr>
<td>ColBest</td>
<td>Col best indicators</td>
<td></td>
</tr>
<tr>
<td>ColContr</td>
<td>Col contributions to inertia</td>
<td></td>
</tr>
<tr>
<td>ColCoors</td>
<td>Col coordinates</td>
<td></td>
</tr>
<tr>
<td>ColProfiles</td>
<td>Col profiles</td>
<td>CP</td>
</tr>
<tr>
<td>ColProfilesPct</td>
<td>Col profiles, percents</td>
<td>CP</td>
</tr>
<tr>
<td>ColQualMassIn</td>
<td>Col quality, mass, inertia</td>
<td></td>
</tr>
<tr>
<td>ColSqCos</td>
<td>Col squared cosines</td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>DF, Chi Square (not displayed)</td>
<td></td>
</tr>
<tr>
<td>Deviations</td>
<td>Observed — expected frequencies</td>
<td>DEVIATIONS</td>
</tr>
<tr>
<td>DeviationsPct</td>
<td>Observed — expected percentages</td>
<td>DEVIATIONS</td>
</tr>
<tr>
<td>Expected</td>
<td>Expected frequencies</td>
<td>EXPECTED</td>
</tr>
<tr>
<td>ExpectedPct</td>
<td>Expected percents</td>
<td>EXPECTED</td>
</tr>
<tr>
<td>Intertias</td>
<td>Inertia decomposition table</td>
<td></td>
</tr>
<tr>
<td>Observed</td>
<td>Observed frequencies</td>
<td>OBSERVED</td>
</tr>
<tr>
<td>ObservedPct</td>
<td>Observed percents</td>
<td>OBSERVED</td>
</tr>
<tr>
<td>RowBest</td>
<td>Row best indicators</td>
<td></td>
</tr>
<tr>
<td>RowConstr</td>
<td>Row contributions to inertia</td>
<td></td>
</tr>
<tr>
<td>RowCoors</td>
<td>Row coordinates</td>
<td></td>
</tr>
<tr>
<td>RowProfiles</td>
<td>Row profiles</td>
<td>RP</td>
</tr>
<tr>
<td>RowProfilesPct</td>
<td>Row profiles, percents</td>
<td>RP</td>
</tr>
<tr>
<td>RowQualMassIn</td>
<td>Row quality, mass, inertia</td>
<td></td>
</tr>
<tr>
<td>RowSqCos</td>
<td>Row squared cosines</td>
<td></td>
</tr>
<tr>
<td>SupColCoors</td>
<td>Supp col coordinates</td>
<td></td>
</tr>
<tr>
<td>SupColProfiles</td>
<td>Sup col profiles</td>
<td>CP</td>
</tr>
<tr>
<td>SupColProfilesPct</td>
<td>Sup col profiles, percents</td>
<td>CP</td>
</tr>
</tbody>
</table>
### Table 10.34 ODS Table Names Produced by the DISCRIM Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Univariate statistics</td>
<td>ANOVA</td>
</tr>
<tr>
<td>AvePostCrossVal</td>
<td>Average posterior probabilities,</td>
<td>POSTERR and</td>
</tr>
<tr>
<td></td>
<td>cross validation</td>
<td>CROSSVALIDATE</td>
</tr>
<tr>
<td>AvePostResub</td>
<td>Average posterior probabilities,</td>
<td>POSTERR</td>
</tr>
<tr>
<td></td>
<td>resubstitution</td>
<td></td>
</tr>
<tr>
<td>AvePostTestClass</td>
<td>Average posterior probabilities,</td>
<td>POSTERR and</td>
</tr>
<tr>
<td></td>
<td>test classification</td>
<td>TEST=</td>
</tr>
<tr>
<td>AveRSquare</td>
<td>Average R-Square</td>
<td>ANOVA</td>
</tr>
<tr>
<td>BCorr</td>
<td>Between-class correlations</td>
<td>BCORR</td>
</tr>
<tr>
<td>BCov</td>
<td>Between-class covariances</td>
<td>BCOV</td>
</tr>
<tr>
<td>BSSCP</td>
<td>Between-class SSCP matrix</td>
<td>BSSCP</td>
</tr>
<tr>
<td>BStruct</td>
<td>Between canonical structure</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>CanCorr</td>
<td>Canonical correlations</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>CanonicalMeans</td>
<td>Class means on canonical variables</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>ChiSq</td>
<td>Chi-Square information</td>
<td>POOL=TEST</td>
</tr>
<tr>
<td>ClassifiedCrossVal</td>
<td>Number of observations and</td>
<td>CROSSVALIDATE</td>
</tr>
<tr>
<td></td>
<td>percent classified, cross</td>
<td></td>
</tr>
<tr>
<td></td>
<td>validation</td>
<td></td>
</tr>
</tbody>
</table>

Table Name | Description | Option |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SupColQuality</td>
<td>Supp col quality</td>
<td></td>
</tr>
<tr>
<td>SupCols</td>
<td>Supplementary col frequency</td>
<td>OBSERVED</td>
</tr>
<tr>
<td>SupColsPct</td>
<td>Supplementary col percents</td>
<td>OBSERVED</td>
</tr>
<tr>
<td>SupColSqCos</td>
<td>Supplementary col squared cosines</td>
<td></td>
</tr>
<tr>
<td>SupRows</td>
<td>Supplementary row frequencies</td>
<td>OBSERVED</td>
</tr>
<tr>
<td>SupRowCoors</td>
<td>Supplementary row coordinates</td>
<td></td>
</tr>
<tr>
<td>SupRowProfiles</td>
<td>Supplementary row profiles</td>
<td>RP</td>
</tr>
<tr>
<td>SupRowProfilesPct</td>
<td>Supplementary row profiles, percents</td>
<td>RP</td>
</tr>
<tr>
<td>SupRowQuality</td>
<td>Supplementary row quality</td>
<td></td>
</tr>
<tr>
<td>SupRowsPct</td>
<td>Supplementary row percents</td>
<td>OBSERVED</td>
</tr>
<tr>
<td>SupRowSqCos</td>
<td>Supplementary row square cosines</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>ClassifiedResub</td>
<td>Number of observations and percent classified, resubstitution</td>
<td></td>
</tr>
<tr>
<td>ClassifiedTestClass</td>
<td>Number of observations and percent classified, test classification</td>
<td>TEST=</td>
</tr>
<tr>
<td>Counts</td>
<td>Number of observations, variables, classes, DF</td>
<td></td>
</tr>
<tr>
<td>CovDF</td>
<td>DF for covariance matrices, not displayed</td>
<td>any *COV option</td>
</tr>
<tr>
<td>Dist</td>
<td>Squared distances</td>
<td>MAHALONOBIS</td>
</tr>
<tr>
<td>DistFValues</td>
<td>F values based on squared distances</td>
<td>MAHALONOBIS</td>
</tr>
<tr>
<td>DistGeneralized</td>
<td>Generalized squared distances</td>
<td></td>
</tr>
<tr>
<td>DistProb</td>
<td>Probabilities for F values from squared distances</td>
<td>MAHALONOBIS</td>
</tr>
<tr>
<td>ErrorCrossVal</td>
<td>Error count estimates, cross validation</td>
<td>CROSSVALIDATE</td>
</tr>
<tr>
<td>ErrorResub</td>
<td>Error count estimates, resubstitution</td>
<td></td>
</tr>
<tr>
<td>ErrorTestClass</td>
<td>Error count estimates, test classification</td>
<td>TEST=</td>
</tr>
<tr>
<td>Levels</td>
<td>Class level information</td>
<td></td>
</tr>
<tr>
<td>LinearDiscFunc</td>
<td>Linear discriminant function</td>
<td>POOL=YES</td>
</tr>
<tr>
<td>LogDet</td>
<td>Log determinant of the covariance matrix</td>
<td></td>
</tr>
<tr>
<td>MultStat</td>
<td>MANOVA</td>
<td>MANOVA</td>
</tr>
<tr>
<td>PCoeff</td>
<td>Pooled standard canonical coefficients</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>PCorr</td>
<td>Pooled within-class correlations</td>
<td>PCORR</td>
</tr>
<tr>
<td>PCov</td>
<td>Pooled within-class covariances</td>
<td>PCOV</td>
</tr>
<tr>
<td>PSSCP</td>
<td>Pooled within-class SSCP matrix</td>
<td>PSSCP</td>
</tr>
<tr>
<td>PStdMeans</td>
<td>Pooled standardized class means</td>
<td>STDMEAN</td>
</tr>
<tr>
<td>PStruc</td>
<td>Pooled within canonical structure</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>PostCrossVal</td>
<td>Posterior probabilities, cross validation</td>
<td>CROSSLIST or CROSSLISTERR</td>
</tr>
<tr>
<td>PostErrCrossVal</td>
<td>Posterior error estimates, cross validation</td>
<td>POSTERR and CROSSVALIDATE</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>PostErrResub</td>
<td>Posterior error estimates, resubstitution</td>
<td>POSTERR</td>
</tr>
<tr>
<td>PostErrTestClass</td>
<td>Posterior error estimates, test classification</td>
<td>POSTERR and TEST=</td>
</tr>
<tr>
<td>PostResub</td>
<td>Posterior probabilities, resubstitution</td>
<td>LIST or LISTERR</td>
</tr>
<tr>
<td>PostTestClass</td>
<td>Posterior probabilities, test classification</td>
<td>TESTLIST or TESTLISTERR</td>
</tr>
<tr>
<td>RCoef</td>
<td>Raw canonical coefficients</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>TCoeff</td>
<td>Total-sample standard canonical coefficients</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>TCorr</td>
<td>Total-sample correlations</td>
<td>TCORR</td>
</tr>
<tr>
<td>TCov</td>
<td>Total-sample covariances</td>
<td>TCOV</td>
</tr>
<tr>
<td>TSSCP</td>
<td>Total-sample SSCP matrix</td>
<td>TSSCP</td>
</tr>
<tr>
<td>TStdMeans</td>
<td>Total standardized class means</td>
<td>STDMEAN</td>
</tr>
<tr>
<td>TStruc</td>
<td>Total canonical structure</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>WCorr</td>
<td>Within-class correlations</td>
<td>WCORR</td>
</tr>
<tr>
<td>WCov</td>
<td>Within-class covariances</td>
<td>WCOV</td>
</tr>
<tr>
<td>WSSCP</td>
<td>Within-class SSCP matrices</td>
<td>WSSCP</td>
</tr>
</tbody>
</table>

**Table 10.35 ODS Table Names Produced by the FACTOR Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlphaCoef</td>
<td>Coefficient alpha for each factor</td>
<td>METHOD=ALPHA</td>
</tr>
<tr>
<td>CanCorr</td>
<td>Squared canonical correlations</td>
<td>METHOD=ML</td>
</tr>
<tr>
<td>CondStdDev</td>
<td>Conditional standard deviations</td>
<td>SIMPLE w/PARTIAL</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td>METHOD=PRINIT, =ALPHA, =ML, or =ULS</td>
</tr>
<tr>
<td>Corr</td>
<td>Correlations</td>
<td>CORR</td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>Eigenvalues</td>
<td>default or SCREE</td>
</tr>
<tr>
<td>Eigenvectors</td>
<td>Eigenvectors</td>
<td>EIGENVECTORS</td>
</tr>
<tr>
<td>FactorWeightRotate</td>
<td>Factor weights for rotation</td>
<td>HKPOWER=</td>
</tr>
<tr>
<td>FactorPattern</td>
<td>Factor pattern</td>
<td>ROTATE= any oblique rotation</td>
</tr>
<tr>
<td>FactorStructure</td>
<td>Factor structure</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>FinalCommun</td>
<td>Final communalities</td>
<td>default</td>
</tr>
<tr>
<td>FinalCommunWgt</td>
<td>Final communalities with weights</td>
<td>METHOD=ML or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>METHOD=ALPHA</td>
</tr>
<tr>
<td>FitMeasures</td>
<td>Measures of fit</td>
<td>METHOD=ML</td>
</tr>
<tr>
<td>ImageCoef</td>
<td>Image coefficients</td>
<td>METHOD=IMAGE</td>
</tr>
<tr>
<td>ImageCov</td>
<td>Image covariance matrix</td>
<td>METHOD=IMAGE</td>
</tr>
<tr>
<td>ImageFactors</td>
<td>Image factor matrix</td>
<td>METHOD=IMAGE</td>
</tr>
<tr>
<td>InputFactorPattern</td>
<td>Input factor pattern</td>
<td>METHOD=PATTERN with PRINT or ALL</td>
</tr>
<tr>
<td>InputScoreCoef</td>
<td>Standardized input scoring coefficients</td>
<td>METHOD=SCORE with PRINT or ALL</td>
</tr>
<tr>
<td>InterFactorCorr</td>
<td>Inter-factor correlations</td>
<td>ROTATE=any oblique rotation</td>
</tr>
<tr>
<td>InvCorr</td>
<td>Inverse correlation matrix</td>
<td>ALL</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>METHOD=PRINIT, =ALPHA, =ML, or =ULS</td>
</tr>
<tr>
<td>MultipleCorr</td>
<td>Squared multiple correlations</td>
<td>METHOD=IMAGE or METHOD=HARRIS</td>
</tr>
<tr>
<td>NormObliqueTrans</td>
<td>Normalized oblique transformation matrix</td>
<td>ROTATE=any oblique rotation</td>
</tr>
<tr>
<td>ObliqueRotFactPat</td>
<td>Rotated factor pattern</td>
<td>ROTATE=any oblique rotation</td>
</tr>
<tr>
<td>ObliqueTrans</td>
<td>Oblique transformation matrix</td>
<td>HKPOWER=</td>
</tr>
<tr>
<td>OrthRotFactPat</td>
<td>Rotated factor pattern</td>
<td>ROTATE=any orthogonal rotation</td>
</tr>
<tr>
<td>OrthTrans</td>
<td>Orthogonal transformational matrix</td>
<td>ROTATE=any orthogonal rotation</td>
</tr>
<tr>
<td>ParCorrControlFactor</td>
<td>Partial correlations controlling factors</td>
<td>RESIDUAL</td>
</tr>
<tr>
<td>ParCorrControlVar</td>
<td>Partial correlations controlling other</td>
<td>MSA</td>
</tr>
<tr>
<td></td>
<td>variables</td>
<td></td>
</tr>
<tr>
<td>PartialCorr</td>
<td>Partial correlations</td>
<td>MSA or CORR w/PARTIAL</td>
</tr>
<tr>
<td>PriorCommunalEst</td>
<td>Prior communality estimates</td>
<td>PRIORS=, METHOD=ML, or METHOD=ALPHA</td>
</tr>
<tr>
<td>ProcrustesTarget</td>
<td>Target matrix for Procrustean transformation</td>
<td>ROTATE=PROCRUSTES or ROTATE=PROMAX</td>
</tr>
<tr>
<td>ProcrustesTrans</td>
<td>Procrustean transformation matrix</td>
<td>ROTATE=PROCRUSTES or ROTATE=PROMAX</td>
</tr>
<tr>
<td>RMSOffDiagPartials</td>
<td>Root mean square off-diagonal partials</td>
<td>RESIDUAL</td>
</tr>
<tr>
<td>RMSOffDiagResids</td>
<td>Root mean square off-diagonal residuals</td>
<td>RESIDUAL</td>
</tr>
<tr>
<td>ReferenceAxisCorr</td>
<td>Reference axis correlations</td>
<td>ROTATE=any oblique rotation</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>ReferenceStructure</td>
<td>Reference structure</td>
<td>ROTATE=any oblique rotation</td>
</tr>
<tr>
<td>ResCorrUniqueDiag</td>
<td>Residual correlations with uniqueness on the diagonal</td>
<td>RESIDUAL</td>
</tr>
<tr>
<td>SamplingAdequacy</td>
<td>Kaiser's measure of sampling adequacy</td>
<td>MSA</td>
</tr>
<tr>
<td>SignifTests</td>
<td>Significance tests</td>
<td>METHOD=ML</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>StdScoreCoef</td>
<td>Standardized scoring coefficients</td>
<td>SCORE</td>
</tr>
<tr>
<td>VarExplain</td>
<td>Variance explained</td>
<td></td>
</tr>
<tr>
<td>VarExplainWgt</td>
<td>Variance explained with weights</td>
<td>METHOD=ML or METHOD=ALPHA</td>
</tr>
<tr>
<td>VarFactorCorr</td>
<td>Squared multiple correlations of the variables with each factor</td>
<td>SCORE</td>
</tr>
<tr>
<td>VarWeightRotate</td>
<td>Variable weights for rotation</td>
<td>NORM=WEIGHT or ROTATE=</td>
</tr>
</tbody>
</table>

**Table 10.36  ODS Table Names Produced by the FASTCLUS Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApproxExpOverAllRSq</td>
<td>Approximate expected overall R-Squared, single number</td>
<td></td>
</tr>
<tr>
<td>CCC</td>
<td>Cubic clustering criterion, single number</td>
<td></td>
</tr>
<tr>
<td>ClusterList</td>
<td>Cluster listing, obs, ID, and distances</td>
<td>LIST</td>
</tr>
<tr>
<td>ClusterSum</td>
<td>Cluster summary, cluster number, distances</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>ClusterCenters</td>
<td>Cluster centers</td>
<td></td>
</tr>
<tr>
<td>ClusterDispersion</td>
<td>Cluster dispersion</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>Criterion</td>
<td>Criterion based on final seeds, single number</td>
<td></td>
</tr>
<tr>
<td>DistBetweenClust</td>
<td>Distance between clusters</td>
<td></td>
</tr>
<tr>
<td>InitialSeeds</td>
<td>Initial seeds</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PROC Statement**
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>IterHistory</td>
<td>Iteration history, various statistics for each iteration</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>MinDist</td>
<td>Minimum distance between initial seeds, single number</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>NumberOfBins</td>
<td>Number of bins</td>
<td></td>
</tr>
<tr>
<td>ObsOverAllRSquare</td>
<td>Observed overall R-Squared. single number</td>
<td>SUMMARY</td>
</tr>
<tr>
<td>PrelScaleEst</td>
<td>Preliminary $L(1)$ scale estimate, single number</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>PseudoFStat</td>
<td>Pseudo F statistic, single number</td>
<td></td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics for input variables</td>
<td></td>
</tr>
<tr>
<td>VariableStat</td>
<td>Statistics for variables within clusters</td>
<td></td>
</tr>
</tbody>
</table>

**Table 10.37  ODS Table Names Produced by the GAM Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANODEV</td>
<td>Analysis of deviance table for smoothing variables</td>
</tr>
<tr>
<td>ClassSummary</td>
<td>Summary of class variables</td>
</tr>
<tr>
<td>InputSummary</td>
<td>Data summary</td>
</tr>
<tr>
<td>IterSummary</td>
<td>Iteration summary</td>
</tr>
<tr>
<td>FitSummary</td>
<td>Fit parameters and fit summary</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimation for regression variables</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PROC Statement**

**ODS Tables Created by the MODEL Statement**

| Iteration          | Iteration history table                        | ITPRINT  |
## Table 10.38 ODS Table Names Produced by the GENMOD Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the CLASS Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClassLevels</td>
<td>Class variable levels</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the CONTRAST Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contrasts</td>
<td>Tests of contrasts</td>
<td></td>
</tr>
<tr>
<td>ContrastCoef</td>
<td>Contrast coefficients</td>
<td>E</td>
</tr>
<tr>
<td>LinDep</td>
<td>Linearly dependent rows of contrasts</td>
<td></td>
</tr>
<tr>
<td>NonEst</td>
<td>Nonestimable rows of contrasts</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the MODEL Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Parameter estimate correlation matrix</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Parameter estimate covariance matrix</td>
<td>COVB</td>
</tr>
<tr>
<td>IterLRCI</td>
<td>Iteration history for likelihood ratio confidence intervals</td>
<td>LRCI ITPRINT</td>
</tr>
<tr>
<td>IterParms</td>
<td>Iteration history for parameter estimates</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>IterType3</td>
<td>Iteration history for Type 3 statistics</td>
<td>TYPE3 ITPRINT</td>
</tr>
<tr>
<td>LRCI</td>
<td>Likelihood ratio confidence intervals</td>
<td>LRCI ITPRINT</td>
</tr>
<tr>
<td>LagrangeStatistics</td>
<td>Lagrange statistics</td>
<td>NOINT or NOSCALE</td>
</tr>
<tr>
<td>LastGradHess</td>
<td>Last evaluation of the gradient and Hessian</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td></td>
</tr>
<tr>
<td>Modelfit</td>
<td>Goodness-of-fit statistics</td>
<td></td>
</tr>
<tr>
<td>ObStats</td>
<td>Observation-wise statistics</td>
<td>OBSTATS, CL, PREDICTED, RESIDUALS, or XVARS</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>ParmInfo</td>
<td>Parameter indices</td>
<td></td>
</tr>
<tr>
<td>ResponseProfiles</td>
<td>Frequency counts for multinomial models</td>
<td>DIST=MULTINOMIAL</td>
</tr>
<tr>
<td>Type1</td>
<td>Type 1 tests</td>
<td>TYPE1</td>
</tr>
<tr>
<td>Type3</td>
<td>Type 3 tests</td>
<td>TYPE3</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>----------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>ODS Tables Created by the ESTIMATE Statement</td>
<td>Estimated of contrasts</td>
<td></td>
</tr>
<tr>
<td>Estimates</td>
<td>Estimates of contrasts</td>
<td></td>
</tr>
<tr>
<td>EstimateCoef</td>
<td>Contrast coefficients</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODS Tables Created by the REPEATED Statement</td>
<td>GEE parameter estimates with empirical standard errors</td>
<td></td>
</tr>
<tr>
<td>GEEEmpPEst</td>
<td>GEE parameter estimates with empirical standard errors</td>
<td></td>
</tr>
<tr>
<td>GEELogORInfo</td>
<td>GEE log odds ratio model information</td>
<td>LOGOR=</td>
</tr>
<tr>
<td>GEEModInfo</td>
<td>GEE model information</td>
<td></td>
</tr>
<tr>
<td>GEEModelPEst</td>
<td>GEE model information</td>
<td>MODELSE</td>
</tr>
<tr>
<td>GEENCorr</td>
<td>GEE model-based correlation matrix</td>
<td>MCORRB</td>
</tr>
<tr>
<td>GEENCov</td>
<td>GEE model-based covariance matrix</td>
<td>MCOVB</td>
</tr>
<tr>
<td>GEERCov</td>
<td>GEE empirical correlation matrix</td>
<td>ECOVB</td>
</tr>
<tr>
<td>GEERCov</td>
<td>GEE empirical covariance matrix</td>
<td></td>
</tr>
<tr>
<td>GEEWCorr</td>
<td>GEE working correlation matrix</td>
<td>CORRW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODS Tables Created by the MODEL CONTRAST Statement</td>
<td>Iteration history for contrasts</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>IterContrasts</td>
<td>Iteration history for contrasts</td>
<td>ITPRINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODS Tables Created by the MODEL REPEATED Statement</td>
<td>Iteration history for GEE parameter estimates</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>IterParmsGEE</td>
<td>Iteration history for GEE parameter estimates</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>LastGEEGrad</td>
<td>Last evaluation of the generalized gradient and Hessian</td>
<td>ITPRINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODS Tables Created by the LSMEANS Statement</td>
<td>Coefficients for least squares means</td>
<td>E</td>
</tr>
<tr>
<td>LSMeanCoef</td>
<td>Coefficients for least squares means</td>
<td>E</td>
</tr>
<tr>
<td>LSMeanDiffs</td>
<td>Least squares means differences</td>
<td>DIFF</td>
</tr>
<tr>
<td>LSMMeans</td>
<td>Least squares means</td>
<td></td>
</tr>
</tbody>
</table>
Table 10.39 ODS Table Names Produced by the GLM Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DependentInfo</td>
<td>Simultaneously analyzed dependent variables</td>
<td>default when there are multiple dependent variables with different patterns of missing values</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>R-Square, C.V., root MSE, and dependent mean</td>
<td></td>
</tr>
<tr>
<td>MatrixRepresentation</td>
<td>X matrix element representation</td>
<td>as needed for other options</td>
</tr>
<tr>
<td>ModelANOVA</td>
<td>ANOVA for model terms</td>
<td></td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations</td>
<td></td>
</tr>
<tr>
<td>OverallANOVA</td>
<td>Overall ANOVA</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the CLASS Statement**

ClassLevels Classification variable levels

**ODS Tables Created by the CONTRAST Statement**

AltErrContrasts ANOVA table for contrasts with alternative error E=

ContrastCoef L matrix for contrast EST

Contrasts ANOVA table for contrasts

**ODS Tables Created by the ESTIMATE Statement**

Estimates Estimate statement result

**ODS Tables Created by the LSMEANS Statement**

Diff PDiff matrix of least-squares means PDIFF

LSMeanCL Confidence interval for LS-means CL

LSMeanCoef Coefficients of least-squares means E

LSMeanDiffCL Confidence interval for LS-mean differences PDIFF and CL

LSMeans Least-squares means

SimDetails Details of difference quantile simulation ADJUST=SIMULATE(REPORT)

SimResults Evaluation of difference quantile simulation ADJUST=SIMULATE(REPORT)

SlicedANOVA Sliced effect ANOVA table SLICE
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the MEANS Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bartlett</td>
<td>Bartlett’s homogeneity of variance test</td>
<td>HOVTEST=BARTLETT</td>
</tr>
<tr>
<td>CLDiffs</td>
<td>Multiple comparisons of pairwise differences</td>
<td>CLDIFF, DUNNETT, or (Unequal cells and not LINES)</td>
</tr>
<tr>
<td>CLDiffsInfo</td>
<td>Information for multiple comparisons of pairwise differences</td>
<td>CLDIFF, DUNNETT, or (Unequal cells and not LINES)</td>
</tr>
<tr>
<td>CLMeans</td>
<td>Multiple comparisons of means with confidence/comparison interval</td>
<td>CLM</td>
</tr>
<tr>
<td>CLMeansInfo</td>
<td>Information for multiple comparison of means with confidence/comparison interval</td>
<td>CLM</td>
</tr>
<tr>
<td>HOVFTest</td>
<td>Homogeneity of variance ANOVA</td>
<td>HOVTEST</td>
</tr>
<tr>
<td>MCLines</td>
<td>Multiple comparisons LINES output</td>
<td>LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (Equal cells and not CLDIFF)</td>
</tr>
<tr>
<td>MCLinesInfo</td>
<td>Information for multiple comparison LINES output</td>
<td>LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (Equal cells and not CLDIFF)</td>
</tr>
<tr>
<td>MCLinesRange</td>
<td>Ranges for multiple range MC tests</td>
<td>LINES, ((DUNCAN, WALLER, SNK, or REGWQ) and not (CLDIFF or CLM)), or (Equal cells and not CLDIFF)</td>
</tr>
<tr>
<td>Means</td>
<td>Group means</td>
<td></td>
</tr>
<tr>
<td>Welch</td>
<td>Welch’s ANOVA</td>
<td>WELCH</td>
</tr>
</tbody>
</table>

<p>| <strong>ODS Tables Created by the MODEL Statement</strong>                                      |
| Aliasng         | Type 1, 2, 3, 4 aliasing structure         | (E1, E2, E3, or E4) and ALIASING |
| EstFunc         | Type 1, 2, 3, 4 estimable functions        | E1, E2, E3, or E4               |
| GAiasng         | General form of aliasing structure         | E and ALIASING                  |
| GEstFunc        | General form of estimable functions        | E                               |
| InvXPX          | Inv(XX) matrix                             | INVERSE                         |</p>
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterEstimates</td>
<td>Estimated linear model coefficients</td>
<td>SOLUTION</td>
</tr>
<tr>
<td>PredictedInfo</td>
<td>Predicted values info</td>
<td>PREDICTED, CLM, or CLI</td>
</tr>
<tr>
<td>PredictedValues</td>
<td>Predicted values</td>
<td>PREDICTED, CLM, or CLI</td>
</tr>
<tr>
<td>Tolerances</td>
<td>XX tolerances</td>
<td>TOLERANCE</td>
</tr>
<tr>
<td>XPX</td>
<td>XX matrix</td>
<td>XPX</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MANOVA or REPEATED Statements**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanAnalysis</td>
<td>Canonical analysis</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>CanCoef</td>
<td>Canonical coefficients</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>CanStructure</td>
<td>Canonical structure</td>
<td>CANONICAL</td>
</tr>
<tr>
<td>ErrorSSCP</td>
<td>Error SSCP matrix</td>
<td>PRINTE</td>
</tr>
<tr>
<td>HypothesisSSCP</td>
<td>Hypothesis SSCP matrix</td>
<td>PRINTE</td>
</tr>
<tr>
<td>PartialCorr</td>
<td>Partial correlation matrix</td>
<td>PRINTE</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MANOVA Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharStruct</td>
<td>Characteristic roots and vectors</td>
<td>not CANONICAL</td>
</tr>
<tr>
<td>MANOVATransform</td>
<td>Multivariate transformation matrix</td>
<td>M=</td>
</tr>
<tr>
<td>MultStat</td>
<td>Multivariate tests</td>
<td></td>
</tr>
<tr>
<td>Tests</td>
<td>Summary ANOVA for specified MANOVA H= effects</td>
<td>H=SUMMARY</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the RANDOM Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpectedMeanSquares</td>
<td>Expected mean squares</td>
<td></td>
</tr>
<tr>
<td>QForm</td>
<td>Quadratic form for expected mean squares</td>
<td>Q</td>
</tr>
<tr>
<td>RandomModelANOVA</td>
<td>Random effect tests</td>
<td>TEST</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the REPEATED Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharStruct</td>
<td>Characteristic roots and vectors</td>
<td>PRINTRV</td>
</tr>
<tr>
<td>Epsilons</td>
<td>Greenhouse-Geisser and Huynh-Feldt epsilons</td>
<td></td>
</tr>
<tr>
<td>RepeatedLevelInfo</td>
<td>Correspondence between dependents and repeated measures levels</td>
<td></td>
</tr>
<tr>
<td>RepeatedTransform</td>
<td>Repeated measures transformation matrix</td>
<td>PRINTM</td>
</tr>
</tbody>
</table>
### Table 10.40 ODS Table Names Produced by the GLMMOD Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DependentInfo</td>
<td>Simultaneously analyzed dependent variables</td>
<td>default when there are multiple dependent variables</td>
</tr>
<tr>
<td>DesignPoints</td>
<td>Design matrix</td>
<td></td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>Parameters and associated column numbers</td>
<td></td>
</tr>
</tbody>
</table>

### ODS Tables Created by the CLASS Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassLevels</td>
<td>Table of class levels</td>
</tr>
</tbody>
</table>

### Table 10.41 ODS Table Names Produced by the INBREED Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvgCovCoef</td>
<td>Averages of covariance coefficient matrix</td>
<td>COVAR and AVERAGE</td>
</tr>
<tr>
<td>AvgInbreedingCoef</td>
<td>Averages of inbreeding coefficient matrix</td>
<td>AVERAGE</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the GENDER Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>MatingCovCoef</td>
<td>Covariance coefficients of matings</td>
<td>COVAR</td>
</tr>
<tr>
<td>MatingInbreedingCoef</td>
<td>Inbreeding coefficients of matings</td>
<td></td>
</tr>
</tbody>
</table>
### ODS Tables Created by the PROC Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CovarianceCoefficient</td>
<td>Covariance coefficient table</td>
<td>COVAR</td>
</tr>
<tr>
<td>InbreedingCoefficient</td>
<td>Inbreeding coefficient table</td>
<td></td>
</tr>
<tr>
<td>IndividualCovCoef</td>
<td>Inbreeding coefficients of</td>
<td>IND and COVAR</td>
</tr>
<tr>
<td></td>
<td>individuals</td>
<td></td>
</tr>
<tr>
<td>IndividualInbreedingCoef</td>
<td>Inbreeding coefficients of</td>
<td>IND</td>
</tr>
<tr>
<td></td>
<td>individuals</td>
<td></td>
</tr>
<tr>
<td>NumberOfObservations</td>
<td>Number of observations</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10.42  ODS Table Names Produced by the KDE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BivariateStatistics</td>
<td>Bivariate statistics</td>
</tr>
<tr>
<td>Controls</td>
<td>Control variables</td>
</tr>
<tr>
<td>Inputs</td>
<td>Input information</td>
</tr>
<tr>
<td>Levels</td>
<td>Levels of density estimate</td>
</tr>
<tr>
<td>Percentiles</td>
<td>Percentiles of data</td>
</tr>
<tr>
<td>Statistics</td>
<td>Basic statistics</td>
</tr>
</tbody>
</table>

### Table 10.43  ODS Table Names Produced by the LATTICE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Analysis of variance</td>
</tr>
<tr>
<td>AdjTreatmentMeans</td>
<td>Adjusted treatment means</td>
</tr>
<tr>
<td>Statistics</td>
<td>Additional statistics</td>
</tr>
</tbody>
</table>

### Table 10.44  ODS Table Names Produced by the LIFEREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS Tables Created by the CLASS Statement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table Name | Description                        | Option
----------|------------------------------------|--------
ClassLevels| Class variable levels              |        

**ODS Tables Created by the MODEL Statement**

- ConvergenceStatus: Convergence status
- CorrB: Parameter estimate correlation matrix
- CovB: Parameter estimate covariance matrix
- IterHistory: Iteration history
- LagrangeStatistics: Lagrange statistics
- LastGrad: Last evaluation of the gradient
- LastHess: Last evaluation of the Hessian
- ModelInfo: Model information
- ParameterEstimates: Parameter estimates
- ParmInfo: Parameter indices
- Type3Analysis: Type 3 tests

**ODS Tables Created by the PROBPLOT Statement**

- EMIterHistory: Iteration history for Turnbull algorithm
- ProbEstimates: Nonparametric CDF estimates
- Turnbull: Probability estimates from Turnbull algorithm

**Table 10.45**  ODS Table Names Produced by the LIFETEST Procedure

Table Name | Description                        | Option
----------|------------------------------------|--------
CensorPlot | Line-printer plot of censored observations | PLOT=(C, S, LS or LLS), METHOD=PL, and LINEPRINTER
CensoredSummary | Number of event and censored observations | METHOD=PL
DensityPlot | Plot of the density | PLOT=(D) and METHOD=LT
HazardPlot | Plot of the hazards function | PLOT=(H) and METHOD=LT
LifetableEstimates | Lifetable survival estimates | METHOD=LT
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLogSurvivalPlot</td>
<td>Plot of the log of the negative log survivor function</td>
<td>PLOT=(LLS)</td>
</tr>
<tr>
<td>LogSurvivalPlot</td>
<td>Plot of the log survivor function</td>
<td>PLOT=(LS)</td>
</tr>
<tr>
<td>Means</td>
<td>Mean and standard error of survival times</td>
<td>METHOD=PL</td>
</tr>
<tr>
<td>ProductLimitEstimates</td>
<td>Product-limit survival estimates</td>
<td>METHOD=PL</td>
</tr>
<tr>
<td>Quartiles</td>
<td>Quartiles of the survival distribution</td>
<td>METHOD=PL</td>
</tr>
<tr>
<td>SurvivalPlot</td>
<td>Plot of the survivor function</td>
<td>PLOT=(S)</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the STRATA Statement**

- **HomStats**: Rank statistics for testing strata homogeneity
- **HomTests**: Tests for strata homogeneity
- **LogHomCov**: Covariance matrix for the log-rank statistics for strata homogeneity
- **WilHomCov**: Covariance matrix for the Wilcoxon statistics for strata homogeneity

**ODS Tables Created by the TEST Statement**

- **LogForStepSeq**: Forward stepwise sequence for the log-rank statistics for association
- **LogTestCov**: Covariance matrix for log-rank statistics for association
- **LogUniChisq**: Univariate Chi-Squares for log-rank statistic for association
- **WilForStepSeq**: Forward stepwise sequence for the log-rank statistics for association
- **WilTestCov**: Covariance matrix for log-rank statistics for association
- **WilUniChiSq**: Univariate Chi-Squares for Wilcoxon statistic for association
Table 10.46  ODS Table Names Produced by the LOESS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitSummary</td>
<td>Specified fit parameters and fit summary</td>
<td></td>
</tr>
<tr>
<td>ScaleDetails</td>
<td>Extent and scaling of the independent variables</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MODEL Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>kdTree</td>
<td>Structure of kd tree used</td>
<td>DETAILS(kdTree)</td>
</tr>
<tr>
<td>ModelSummary</td>
<td>Summary of all models evaluated</td>
<td>DETAILS(ModelSummary)</td>
</tr>
<tr>
<td>OutputStatistics</td>
<td>Coordinates and fit results at input data points</td>
<td>DETAILS(OutputStatistics)</td>
</tr>
<tr>
<td>PredAtVertices</td>
<td>Coordinates and fitted values at kd tree vertices</td>
<td>DETAILS(PredAtVertices)</td>
</tr>
<tr>
<td>SmoothingCriterion</td>
<td>Criterion value and selected smoothing parameter</td>
<td>SELECT</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the SCORE Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScoreResults</td>
<td>Coordinates and fit results at scoring points</td>
<td>PRINT</td>
</tr>
</tbody>
</table>

Table 10.47  ODS Table Names Produced by the LOGISTIC Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
</table>

**ODS Tables Created by the CONTRAST Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContrastCoeff</td>
<td>L matrix from CONTRAST</td>
<td>E</td>
</tr>
<tr>
<td>ContrastEstimate</td>
<td>Estimates from CONTRAST</td>
<td>ESTIMATE=</td>
</tr>
<tr>
<td>ContrastTest</td>
<td>Wald test for CONTRAST</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the EXACT Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExactOddsRatio</td>
<td>Exact odds ratio</td>
<td>ESTIMATE=ODDS or ESTIMATE=BOTH</td>
</tr>
<tr>
<td>ExactParmEst</td>
<td>Parameter estimates</td>
<td>ESTIMATE=, ESTIMATE=PARM, or ESTIMATE=BOTH</td>
</tr>
<tr>
<td>ExactTests</td>
<td>Conditional exact tests</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>SuffStats</td>
<td>Sufficient statistics</td>
<td>OUTDIST=</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MODEL Statement**

- **Association**: Association of predicted probabilities and observed responses
- **BestSubsets**: Best subset selection
- **ClassLevelInfo**: CLASS variable levels and design variables
- **Classification**: Classification table
- **CLOddsPL**: Profile likelihood confidence limits for odds ratios
- **CLOddsWald**: Wald's confidence limits for odds ratios
- **CLParmPL**: Profile likelihood confidence limits for parameters
- **CLParmWald**: Wald's confidence limits for parameters
- **ConvergenceStatus**: Convergence status
- **CorrB**: Estimated correlation matrix of parameter estimators
- **CovB**: Estimated covariance matrix of parameter estimators
- **CumulativeModelTest**: Test of the cumulative model assumption
- **EffectNotInModel**: Test for effects not in model
- **FastElimination**: Fast backward elimination
- **FitStatistics**: Model fit statistics
- **GlobalScore**: Global score test
- **GlobalTests**: Test for global null hypothesis
- **GoodnessOfFit**: Pearson and deviance goodness-of-fit tests
- **IndexPlots**: Batch capture of the index plots
- **Influence**: Regression diagnostics
- **IterHistory**: Iteration history
- **LackFitChiSq**: Hosmer-Lemeshow Chi-Square test results
- **LackFItPartition**: Partition for the Hosmer-Lemeshow test
- **LastGradient**: Last evaluation of gradient
### Table Name Description Option

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLikeChange</td>
<td>Final change in the log likelihood</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>ModelBuildingSummary</td>
<td>Summary of model building</td>
<td>SELECTION=B, F, or S</td>
</tr>
<tr>
<td>OddsRatios</td>
<td>Odds ratios</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Maximum likelihood estimates</td>
<td></td>
</tr>
<tr>
<td>RSquare</td>
<td>R-Square</td>
<td>RSQUARE</td>
</tr>
<tr>
<td>ResidualChiSq</td>
<td>Residual Chi-Square</td>
<td>SELECTION=F or B</td>
</tr>
<tr>
<td>Type3</td>
<td>Type 3 tests of effects</td>
<td>default (with CLASS variables)</td>
</tr>
</tbody>
</table>

#### ODS Tables Created by the PROC Statement

- **ClassFreq**
  - Frequency breakdown of CLASS variables
  - SIMPLE
- **ClassWgt**
  - Weight breakdown of CLASS variables
  - SIMPLE
- **ModelInfo**
  - Model information
- **ResponseProfile**
  - Response profile
- **SimpleStatistics**
  - Summary statistics for explanatory variables
  - SIMPLE

#### ODS Tables Created by the STRATA Statement

- **StrataSummary**
  - Number of strata with specific response frequencies
- **StrataInfo**
  - Event and non-event frequencies for each stratum
  - INFO

#### ODS Tables Created by the TEST Statement

- **TestPrint1**
  - $L|\text{cov}(b)|L'$ and $Lb-c$
  - PRINT
- **TestPrint2**
  - $\text{Ginv}(L|\text{cov}(b)|L')$ and $\text{Ginv}(L|\text{cov}(b)|L')\langle Lb-c|$ $\rangle$
  - PRINT
- **TestStmts**
  - Linear hypothesis testing results

#### ODS Tables Created by the WEIGHT Statement

- **ClassWgt**
  - Weight breakdown of CLASS variables
  - SIMPLE
Table 10.48 ODS Table Names Produced by the MDS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>DimensionCoef</td>
<td>Dimension coefficients</td>
<td>PCOEF w/COEF= not IDENTITY</td>
</tr>
<tr>
<td>FitMeasures</td>
<td>Measures of fit</td>
<td>PFIT</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td></td>
</tr>
<tr>
<td>PConfig</td>
<td>Estimated coordinates of the objects in the configuration</td>
<td>PCONFIG</td>
</tr>
<tr>
<td>PData</td>
<td>Data matrices</td>
<td>PDATA</td>
</tr>
<tr>
<td>PInAvData</td>
<td>Initial sum of weights and weighted average of data matrices with INAV=DATA</td>
<td>PINAVDATA</td>
</tr>
<tr>
<td>PInEigval</td>
<td>Initial eigenvalues</td>
<td>PINEIGVAL</td>
</tr>
<tr>
<td>PInEigvec</td>
<td>Initial eigenvectors</td>
<td>PINEIGVEC</td>
</tr>
<tr>
<td>PInWeight</td>
<td>Initialization weights</td>
<td>PINWEIGHT</td>
</tr>
<tr>
<td>Transformations</td>
<td>Transformation parameters</td>
<td>PTRANS w/LEVEL=RATIO, INTERVAL, or LOGINTERVAL</td>
</tr>
</tbody>
</table>

Table 10.49 ODS Table Names Produced by the MI Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corr</td>
<td>Pairwise correlations</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>MissPattern</td>
<td>Missing data patterns</td>
<td></td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>Univariate</td>
<td>Univariate statistics</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>VarianceInfo</td>
<td>Between, within, and total variances</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the EM Statement**

- EMEstimates: EM (MLE) estimates
- EMInitEstimates: EM initial estimates
- EMIterHistory: EM (MLE) iteration history

**ODS Tables Created by the MCMC Statement**
### ODS Tables Created by the MONOTONE Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonoDiscrim</td>
<td>Discriminant model group means</td>
<td>DISCRIM (/DETAILS)</td>
</tr>
<tr>
<td>MonoLogistic</td>
<td>Logistic model</td>
<td>LOGISTIC (/DETAILS)</td>
</tr>
<tr>
<td>MonoModel</td>
<td>Multiple monotone models</td>
<td></td>
</tr>
<tr>
<td>MonoPropensity</td>
<td>Propensity score model logistic function</td>
<td>PROPENSITY (/DETAILS)</td>
</tr>
<tr>
<td>MonoReg</td>
<td>Regression model</td>
<td>REG (/DETAILS)</td>
</tr>
<tr>
<td>MonoRegPPM</td>
<td>Predicted mean matching model</td>
<td>REGPMM (/DETAILS)</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the TRANSFORM Statement

<table>
<thead>
<tr>
<th>Transform</th>
<th>Variable transformations</th>
</tr>
</thead>
</table>

### Table 10.50  ODS Table Names Produced by the MIANALYZE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCov</td>
<td>Between-imputation covariance matrix</td>
<td>BCOV</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td></td>
</tr>
<tr>
<td>MultStat</td>
<td>Multivariate inference</td>
<td>MULT</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>TCov</td>
<td>Total covariance matrix</td>
<td>TCOV</td>
</tr>
<tr>
<td>VarianceInfo</td>
<td>Variance information</td>
<td></td>
</tr>
<tr>
<td>WCov</td>
<td>Within-imputation covariance matrix</td>
<td>WCOV</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the TEST Statement

| TestBCov       | Between-imputation covariance matrix for $L_{ij}$ | BCOV     |
## Chapter 10

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestMultStat</td>
<td>Multivariate inference for $L_\beta$</td>
<td>MULT</td>
</tr>
<tr>
<td>TestParameterEstimates</td>
<td>Parameter estimates for $L_\beta$</td>
<td></td>
</tr>
<tr>
<td>TestSpec</td>
<td>Test specification, $L$ and $c$</td>
<td></td>
</tr>
<tr>
<td>TestTCov</td>
<td>Total covariance matrix for $L_\beta$</td>
<td>TCOV</td>
</tr>
<tr>
<td>TestVarianceInfo</td>
<td>Variance information for $L_\beta$</td>
<td>WCOV</td>
</tr>
<tr>
<td>TestWCov</td>
<td>Within—imputation covariance matrix for $L_\beta$</td>
<td></td>
</tr>
</tbody>
</table>

**Table 10.51 ODS Table Names Produced by the MODECLUS Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundaryFreq</td>
<td>Boundary objects information</td>
<td>BOUNDARY (or ALL)</td>
</tr>
<tr>
<td>ClusterList</td>
<td>Cluster listing, cluster ID, frequency, density, etc.</td>
<td>LIST (or ALL)</td>
</tr>
<tr>
<td>ClusterStats</td>
<td>Cluster statistics</td>
<td></td>
</tr>
<tr>
<td>ClusterStats</td>
<td>Cluster statistics, significance test statistics</td>
<td>TEST or JOIN (or ALL)</td>
</tr>
<tr>
<td>ClusterSummary</td>
<td>Cluster summary</td>
<td>CROSS or CROSSLIST (or ALL)</td>
</tr>
<tr>
<td>ClusterSummary</td>
<td>Cluster summary, crossvalidation criterion</td>
<td></td>
</tr>
<tr>
<td>ClusterSummary</td>
<td>Cluster summary, clusters joined information</td>
<td>JOIN (or ALL)</td>
</tr>
<tr>
<td>CrossList</td>
<td>Cross-validated log density</td>
<td>CROSSLIST</td>
</tr>
<tr>
<td>ListLocal</td>
<td>Local dimensionality estimates</td>
<td>LOCAL</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>Nearest neighbor list</td>
<td>NEIGHBOR (or ALL)</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>SIMPLE (or ALL)</td>
</tr>
<tr>
<td>Trace</td>
<td>Trace of clustering algorithm (METHOD=6 only)</td>
<td>TRACE (or ALL) with METHOD=6</td>
</tr>
<tr>
<td>UnassignObjects</td>
<td>Information on unassigned objects</td>
<td>LIST (or ALL)</td>
</tr>
</tbody>
</table>
### Table 10.52 ODS Table Names Produced by the MULTTEST Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Continuous variable</td>
<td>TEST with MEAN</td>
</tr>
<tr>
<td></td>
<td>tabulations</td>
<td></td>
</tr>
<tr>
<td>Contrasts</td>
<td>Contrast coefficients</td>
<td></td>
</tr>
<tr>
<td>Discrete</td>
<td>Discrete variable tabulations</td>
<td>TEST with CA, FT, PETO, or FISHER</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td></td>
</tr>
<tr>
<td>pValues</td>
<td>p-values from the tests</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10.53 ODS Table Names Produced by the NESTED Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANCOVA</td>
<td>Analysis of covariance</td>
</tr>
<tr>
<td>ANOVA</td>
<td>Analysis of variance</td>
</tr>
<tr>
<td>EMSCoef</td>
<td>Coefficients of expected mean squares</td>
</tr>
<tr>
<td>Statistics</td>
<td>Overall statistics for fit</td>
</tr>
</tbody>
</table>

### Table 10.54 ODS Table Names Produced by the NLIN Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Analysis of variance</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation of the parameters</td>
</tr>
<tr>
<td>EstSummary</td>
<td>Summary of the estimation</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration output</td>
</tr>
<tr>
<td>MissingValues</td>
<td>Missing values generated by the program</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the LIST Statement**

| ProgList              | List of the compiled program |
## ODS Tables Created by the LISTCODE Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeList</td>
<td>List of program statements</td>
</tr>
</tbody>
</table>

## ODS Tables Created by the LISTDEP Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeDependency</td>
<td>Variable cross reference</td>
</tr>
</tbody>
</table>

## ODS Tables Created by the LISTDER Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstDerivatives</td>
<td>First derivative table</td>
</tr>
</tbody>
</table>

### Table 10.55 ODS Table Names Produced by the NLMIXED Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdditionalEstimates</td>
<td>Results from ESTIMATE statements</td>
<td>ESTIMATE</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>CorrMatAddEst</td>
<td>Correlation matrix of additional estimates</td>
<td>ECORR</td>
</tr>
<tr>
<td>CorrMatParmEst</td>
<td>Correlation matrix of parameter estimates</td>
<td>CORR</td>
</tr>
<tr>
<td>CovMatAddEst</td>
<td>Covariance matrix of additional estimates</td>
<td>ECOV</td>
</tr>
<tr>
<td>CovMatParmEst</td>
<td>Covariance matrix of parameter estimates</td>
<td>COV</td>
</tr>
<tr>
<td>DerAddEst</td>
<td>Derivatives of additional estimates</td>
<td>EDER</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Dimensions of the problem</td>
<td></td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td></td>
</tr>
<tr>
<td>Hessian</td>
<td>Second derivative matrix</td>
<td>HESS</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>Specifications</td>
<td>Model specifications</td>
<td></td>
</tr>
<tr>
<td>StartingHessian</td>
<td>Starting hessian matrix</td>
<td>START HESS</td>
</tr>
<tr>
<td>StartingValues</td>
<td>Starting values and gradient</td>
<td>START</td>
</tr>
</tbody>
</table>
Table 10.56  ODS Table Names Produced by the NPAR1WAY Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the EXACT Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABMC</td>
<td>Monte Carlo estimates for the Ansari-Bradley exact test</td>
<td>AB or MC</td>
</tr>
<tr>
<td>DataScoresMC</td>
<td>Monte Carlo estimates for the exact test based on data scores</td>
<td>SCORES=DATA or MC</td>
</tr>
<tr>
<td>KlotzMC</td>
<td>Monte Carlo estimates for the Klotz exact test</td>
<td>KLOTZ or MC</td>
</tr>
<tr>
<td>KolSmirExactTest</td>
<td>Kolmogorov-Smirnov exact test</td>
<td>KS</td>
</tr>
<tr>
<td>KruskalWallisMC</td>
<td>Monte Carlo estimates for the Kruskal-Wallis exact test</td>
<td>WILCOXON or MC</td>
</tr>
<tr>
<td>KSMC</td>
<td>Monte Carlo estimates for the Kolmogorov-Smirnov exact test</td>
<td>KS or MC</td>
</tr>
<tr>
<td>MedianMC</td>
<td>Monte Carlo estimates for the median exact test</td>
<td>MEDIAN or MC</td>
</tr>
<tr>
<td>MoodMC</td>
<td>Monte Carlo estimates for the Mood exact test</td>
<td>MOOD or MC</td>
</tr>
<tr>
<td>SavageMC</td>
<td>Monte Carlo estimates for the Savage exact test</td>
<td>SAVAGE or MC</td>
</tr>
<tr>
<td>STMC</td>
<td>Monte Carlo estimates for the Siegel-Tukey one-way analysis</td>
<td>ST or MC</td>
</tr>
<tr>
<td>VWMC</td>
<td>Monte Carlo estimates for the Van der Waerden exact test</td>
<td>VW or MC</td>
</tr>
<tr>
<td>WilcoxonMC</td>
<td>Monte Carlo estimates for the Wilcoxon two-sample exact test</td>
<td>WILCOXON or MC</td>
</tr>
<tr>
<td><strong>ODS Tables Created by the PROC Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANOVA</td>
<td>Analysis of variance</td>
<td>ANOVA</td>
</tr>
<tr>
<td>ABAAnalysis</td>
<td>Ansari-Bradley one-way analysis</td>
<td>AB</td>
</tr>
<tr>
<td>ABScores</td>
<td>Ansari-Bradley scores</td>
<td>AB</td>
</tr>
<tr>
<td>ABTest</td>
<td>Ansari-Bradley two-sample test</td>
<td>AB</td>
</tr>
<tr>
<td>ClassMeans</td>
<td>Class means</td>
<td>ANOVA</td>
</tr>
<tr>
<td>CVMStats</td>
<td>Cramer-von Mises statistics</td>
<td>EDF</td>
</tr>
<tr>
<td>CVMTest</td>
<td>Cramer-von Mises test</td>
<td>EDF</td>
</tr>
<tr>
<td>DataScores</td>
<td>Data scores</td>
<td>SCORES=DATA</td>
</tr>
<tr>
<td>DataScoresAnalysis</td>
<td>Data scores one-way analysis</td>
<td>SCORES=DATA</td>
</tr>
<tr>
<td>DataScoresTest</td>
<td>Data scores two-sample test</td>
<td>SCORES=DATA</td>
</tr>
<tr>
<td>KlotzAnalysis</td>
<td>Klotz one-way analysis</td>
<td>KLOTZ</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>KlotzScores</td>
<td>Klotz scores</td>
<td></td>
</tr>
<tr>
<td>KlotzTest</td>
<td>Klotz two-sample test</td>
<td>KLOTZ</td>
</tr>
<tr>
<td>KolSmir2Stats</td>
<td>Kolmogorov-Smirnov two-sample statistics</td>
<td>EDF</td>
</tr>
<tr>
<td>KolSmirStats</td>
<td>Kolmogorov-Smirnov statistics</td>
<td>EDF</td>
</tr>
<tr>
<td>KolSmirTest</td>
<td>Kolmogorov-Smirnov test</td>
<td>EDF</td>
</tr>
<tr>
<td>KruskalWallisTest</td>
<td>Kruskal-Wallis test</td>
<td>WILCOXON</td>
</tr>
<tr>
<td>KuiperStats</td>
<td>Kuiper two-sample statistics</td>
<td>EDF</td>
</tr>
<tr>
<td>KuiperTest</td>
<td>Kuiper test</td>
<td>EDF</td>
</tr>
<tr>
<td>MedianAnalysis</td>
<td>Median one-way analysis</td>
<td>MEDIAN</td>
</tr>
<tr>
<td>MedianScores</td>
<td>Median scores</td>
<td>MEDIAN</td>
</tr>
<tr>
<td>MedianTest</td>
<td>Median two-sample test</td>
<td>MEDIAN</td>
</tr>
<tr>
<td>MoodAnalysis</td>
<td>Mood one-way analysis</td>
<td>MOOD</td>
</tr>
<tr>
<td>MoodScores</td>
<td>Mood scores</td>
<td>MOOD</td>
</tr>
<tr>
<td>MoodTest</td>
<td>Mood two-sample test</td>
<td>MOOD</td>
</tr>
<tr>
<td>SavageAnalysis</td>
<td>Savage one-way analysis</td>
<td>SAVAGE</td>
</tr>
<tr>
<td>SavageScores</td>
<td>Savage scores</td>
<td>SAVAGE</td>
</tr>
<tr>
<td>SavageTest</td>
<td>Savage two-sample test</td>
<td>SAVAGE</td>
</tr>
<tr>
<td>STAnalysis</td>
<td>Siegel-Tukey one-way analysis</td>
<td>ST</td>
</tr>
<tr>
<td>STScores</td>
<td>Siegel-Tukey scores</td>
<td>ST</td>
</tr>
<tr>
<td>STTest</td>
<td>Siegel-Tukey two-sample test</td>
<td>ST</td>
</tr>
<tr>
<td>VWAnalysis</td>
<td>Van der Waerden one-way analysis</td>
<td>VW</td>
</tr>
<tr>
<td>VWScores</td>
<td>Van der Waerden scores</td>
<td>VW</td>
</tr>
<tr>
<td>VWTest</td>
<td>Van der Waerden two-sample test</td>
<td>VW</td>
</tr>
<tr>
<td>WilcoxonScores</td>
<td>Wilcoxon scores</td>
<td>WILCOXON</td>
</tr>
<tr>
<td>WilcoxonTest</td>
<td>Wilcoxon two-sample test</td>
<td>WILCOXON</td>
</tr>
</tbody>
</table>

Table 10.57  ODS Table Names Produced by the ORTHOREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Analysis of variance</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Overall statistics for fit</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ODS Tables Created by the CLASS Statement</strong></td>
<td></td>
</tr>
<tr>
<td>Levels</td>
<td>Table of class levels</td>
</tr>
</tbody>
</table>

**Table 10.58** ODS Table Names Produced by the PHREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the MODEL Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BestSubsets</td>
<td>Best subset selection</td>
<td>SELECTION=SCORE</td>
</tr>
<tr>
<td>CensoredSummary</td>
<td>Summary of event and censored observations</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Estimated correlation matrix of parameter estimators</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Estimated covariance matrix of parameter estimators</td>
<td>COVB</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Model fit statistics</td>
<td></td>
</tr>
<tr>
<td>GlobalScore</td>
<td>Global Chi-Square test</td>
<td>NOFIT</td>
</tr>
<tr>
<td>GlobalTests</td>
<td>Tests of the global null hypothesis</td>
<td></td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>LastGradient</td>
<td>Last evaluation of gradient</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>ModelBuildingSummary</td>
<td>Summary of model building</td>
<td>SELECTION=B, F, or S</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Maximum likelihood estimates of model parameters</td>
<td></td>
</tr>
<tr>
<td>ResidualChiSq</td>
<td>Residual Chi-Square</td>
<td>SELECTION=F or B</td>
</tr>
<tr>
<td>VariablesNotInModel</td>
<td>Analysis of variables not in the model</td>
<td>SELECTION=F or S</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PROC Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td></td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Summary statistics for explanatory variables</td>
<td>SIMPLE</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the TEST Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TestAverage</td>
<td>Average effect for test</td>
<td>AVERAGE</td>
</tr>
</tbody>
</table>
### Table 10.59  ODS Table Names Produced by the PLAN Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestCoeff</td>
<td>Coefficients for linear hypothesis</td>
<td>E</td>
</tr>
<tr>
<td>TestPrint1</td>
<td>L(\text{cov}(b)L') and Lb-c</td>
<td>PRINT</td>
</tr>
<tr>
<td>TestPrint2</td>
<td>Ginv(L(\text{cov}(b)L') and Ginv(L(\text{cov}(b)L')(Lb-c)</td>
<td>PRINT</td>
</tr>
<tr>
<td>TestStmts</td>
<td>Linear hypotheses testing results</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the FACTOR and TREATMENT Statements**

- PFInfo: Plot factor information
- TFInfo: Treatment factor information

**ODS Tables Created by the FACTOR and no TREATMENT Statements**

- FInfo: General factor information

### Table 10.60  ODS Table Names Produced by the PLS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CenScaleParms</td>
<td>Parameter estimates for centered and scaled data</td>
<td>SOLUTION</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates for raw data</td>
<td>SOLUTION</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MODEL Statement**

- CVResults: Results of cross validation | CV
- CodedCoef: Coded coefficients | DETAILS
- PercentVariation: Variation accounted for by each factor
## Table 10.61  ODS Table Names Produced by the PRINCOMP Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corr</td>
<td>Correlation matrix</td>
<td>default unless COV is specified</td>
</tr>
<tr>
<td>Cov</td>
<td>Covariance matrix</td>
<td>default if COV is specified</td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>Eigenvectors</td>
<td></td>
</tr>
<tr>
<td>Eigenvectors</td>
<td>Eigenvectors</td>
<td></td>
</tr>
<tr>
<td>NObsNVar</td>
<td>Number of observations, variables, and (partial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>variables</td>
<td></td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td></td>
</tr>
<tr>
<td>TotalVariance</td>
<td>Total variance</td>
<td>COV</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>ParCorr</td>
<td>Partial correlation matrix</td>
<td></td>
</tr>
<tr>
<td>ParCov</td>
<td>Uncorrected partial covariance matrix</td>
<td>COV</td>
</tr>
<tr>
<td>RegCoef</td>
<td>Regression coefficients</td>
<td>COV</td>
</tr>
<tr>
<td>RSquareRMSE</td>
<td>Regression statistics: R-Squares and RMSEs</td>
<td></td>
</tr>
<tr>
<td>StdRegCoef</td>
<td>Standardized regression coefficients</td>
<td></td>
</tr>
</tbody>
</table>
### Table 10.62  ODS Table Names Produced by the PRINQUAL Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>Footnotes</td>
<td>Iteration history footnotes</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PROC Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>MAC iteration history</td>
<td>METHOD=MAC</td>
</tr>
<tr>
<td>MGV</td>
<td>MGV iteration history</td>
<td>METHOD=MGV</td>
</tr>
<tr>
<td>MTV</td>
<td>MTV iteration history</td>
<td>METHOD=MTV</td>
</tr>
</tbody>
</table>

### Table 10.63  ODS Table Names Produced by the PROBIT Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassLevels</td>
<td>Class variable levels</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the CLASS Statement**

**ODS Tables Created by the MODEL Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Parameter estimate correlation matrix</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Parameter estimate covariance matrix</td>
<td>COVB</td>
</tr>
<tr>
<td>CovTolerance</td>
<td>Covariance matrix for location and scale</td>
<td></td>
</tr>
<tr>
<td>GoodnessOfFit</td>
<td>Goodness of fit tests</td>
<td>LACKFIT</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>LagrangeStatistics</td>
<td>Lagrange statistics</td>
<td>NOINT</td>
</tr>
<tr>
<td>LastGrad</td>
<td>Last evaluation of the gradient</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>LastHess</td>
<td>Last evaluation of the Hessian</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>LogProbitAnalysis</td>
<td>Probit analysis for log dose</td>
<td>INVERSECL</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td></td>
</tr>
<tr>
<td>MuSigma</td>
<td>Location and scale</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>ParmInfo</td>
<td>Parameter indices</td>
<td></td>
</tr>
<tr>
<td>ProbitAnalysis</td>
<td>Probit analysis for linear dose</td>
<td>INVERSECL</td>
</tr>
</tbody>
</table>
### Table Name | Description | Option
--- | --- | ---
ResponseLevels | Response-covariate profile | LACKFIT
ResponseProfiles | Counts for ordinal data | ALL or ACOV
Type3Analysis | Type 3 tests | ALL or ACOV

### Table 10.64  ODS Table Names Produced by the REG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACovEst</td>
<td>Consistent covariance of estimates matrix</td>
<td>ALL or ACOV</td>
</tr>
<tr>
<td>ANOVA</td>
<td>Model ANOVA table</td>
<td>COLLIN</td>
</tr>
<tr>
<td>CollinDiag</td>
<td>Collinearity diagnostics table</td>
<td>COLLININT</td>
</tr>
<tr>
<td>CollinDiagNoInt</td>
<td>Collinearity diagnostics for no intercept model</td>
<td>COLLININT</td>
</tr>
<tr>
<td>ConditionBounds</td>
<td>Bounds on condition number</td>
<td>(SELECTION=BACKWARD, FORWARD, STEPWISE, MAXR, or MINR) and DETAILS</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation of estimates</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of estimates</td>
<td>COVB</td>
</tr>
<tr>
<td>CrossProducts</td>
<td>Bordered model XX matrix</td>
<td>ALL or XPX</td>
</tr>
<tr>
<td>DWStatistic</td>
<td>Durbin-Watson statistic</td>
<td>ALL or DW</td>
</tr>
<tr>
<td>DependenceEquations</td>
<td>Linear dependence equations</td>
<td></td>
</tr>
<tr>
<td>EntryStatistics</td>
<td>Entry statistics for selection methods</td>
<td>(SELECTION=BACKWARD, FORWARD, STEPWISE, MAXR, or MINR) and DETAILS</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Model fit statistics</td>
<td></td>
</tr>
<tr>
<td>InvXPX</td>
<td>Bordered XX inverse matrix</td>
<td>I</td>
</tr>
<tr>
<td>OutputStatistics</td>
<td>Output statistics table</td>
<td>ALL, CLI, CLM, INFLUENCE, P, or R</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Model parameter estimates</td>
<td></td>
</tr>
<tr>
<td>RemovalStatistics</td>
<td>Removal statistics for selection methods</td>
<td>(SELECTION=BACKWARD, STEPWISE, MAXR, or MINR) and DETAILS</td>
</tr>
<tr>
<td>ResidualStatistics</td>
<td>Residual statistics and PRESS statistic</td>
<td>ALL, CLI, CLM, INFLUENCE, P, or R</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>SelParmEst</td>
<td>Parameter estimates for selection methods</td>
<td>SELECTION=BACKWARD,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORWARD, STEPWISE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAXR, or MINR</td>
</tr>
<tr>
<td>SelectionSummary</td>
<td>Selection summary for forward, backward, and stepwise methods</td>
<td>SELECTION=BACKWARD, FORWARD, or STEPWISE</td>
</tr>
<tr>
<td>SeqParmEst</td>
<td>Sequential parameter estimates</td>
<td>SEQB</td>
</tr>
<tr>
<td>SpecTest</td>
<td>White’s heteroscedasticity test</td>
<td>ALL or SPEC</td>
</tr>
<tr>
<td>SubsetSelSummary</td>
<td>Selection summary for R-Square, adj-RSq, and Cp methods</td>
<td>SELECTION=RSQUARE, ADJRSQ, or CP</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the MTEST Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCorr</td>
<td>Canonical correlations for hypothesis combinations</td>
<td>CANPRINT</td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>MTest eigenvalues</td>
<td>CANPRINT</td>
</tr>
<tr>
<td>Eigenvectors</td>
<td>MTest eigenvectors</td>
<td>CANPRINT</td>
</tr>
<tr>
<td>ErrorPlusHypothesis</td>
<td>MTest error plus hypothesis matrix H+E</td>
<td>PRINT</td>
</tr>
<tr>
<td>ErrorSSCP</td>
<td>MTest error matrix E</td>
<td>PRINT</td>
</tr>
<tr>
<td>HypothesisSSCP</td>
<td>MTest hypothesis matrix</td>
<td>PRINT</td>
</tr>
<tr>
<td>InvMTestCov</td>
<td>Inv(L Ginv(X'X)L') and Inv(Lb-c)</td>
<td>DETAILS</td>
</tr>
<tr>
<td>MTestCov</td>
<td>L Ginv(XX) L' and Lb-c</td>
<td>DETAILS</td>
</tr>
<tr>
<td>MTransform</td>
<td>MTest matrix M, acrossdependents</td>
<td>DETAILS</td>
</tr>
<tr>
<td>MultStat</td>
<td>Multivariate test statistics</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PROC Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corr</td>
<td>Correlation matrix for analysis variables</td>
<td>ALL or CORR</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics for analysis variables</td>
<td>ALL or SIMPLE</td>
</tr>
<tr>
<td>USSCP</td>
<td>Uncorrected SSCP matrix for analysis variables</td>
<td>ALL or USSCP</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the TEST Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACovTestANOVA</td>
<td>Test ANOVA using ACOV estimates</td>
<td>ACOV (MODEL statement)</td>
</tr>
</tbody>
</table>
### Table 10.65  ODS Table Names Produced by the ROBUSTREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvTestCov</td>
<td>( \text{Inv}(L \text{ Ginv}(XX)L') ) and ( \text{Inv}(Lb-c) )</td>
<td>PRINT</td>
</tr>
<tr>
<td>TestANOVA</td>
<td>Test ANOVA table</td>
<td></td>
</tr>
<tr>
<td>TestCov</td>
<td>( L \text{ Ginv}(XX)L' ) and ( Lb-c )</td>
<td>PRINT</td>
</tr>
</tbody>
</table>

#### ODS Tables Created by the CLASS Statement
- ClassLevels: Class variable levels

#### ODS Tables Created by the MODEL Statement
- CorrB: Parameter estimate correlation matrix (CORRB)
- CovB: Parameter estimate covariance matrix (COVB)
- Diagnostics: Outlier diagnostics (DIAGNOSTICS)
- DiagSummary: Summary of the outlier diagnostics
- GoodFit: R², deviance, AIC, and BIC
- ModelInfo: Model information
- ParameterEstimates: Parameter estimates
- ParmInfo: Parameter indices
- SummaryStatistics: Summary statistics for model variables

#### ODS Tables Created by the PROC Statement
- BestEstimates: Best final estimates for LTS (SUBANALYSIS)
- BestSubEstimates: Best estimates for each subgroup (SUBANALYSIS)
- BiasTest: Bias test for MM estimation (BIASTEST)
- CStep: C-Step for LTS fitting (SUBANALYSIS)
- Groups: Groups for LTS fitting (SUBANALYSIS)
- InitLTSProfile: Profile for initial LTS estimate (METHOD)
- InitSProfile: Profile for initial S estimate (METHOD)
- LTSEstimates: LTS parameter estimates (METHOD)
- LTSLocationScale: Location and scale for LTS (METHOD)
### ODS Tables Created by the TEST Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterEstimatesR</td>
<td>Reduced parameter estimates</td>
<td></td>
</tr>
<tr>
<td>TestsProfile</td>
<td>Results for tests</td>
<td></td>
</tr>
</tbody>
</table>

**Table 10.66**  ODS Table Names Produced by the RSREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>Coding coefficients for the independent variables</td>
</tr>
<tr>
<td>ErrorANOVA</td>
<td>Error analysis of variance</td>
</tr>
<tr>
<td>FactorANOVA</td>
<td>Factor analysis of variance</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Overall statistics for fit</td>
</tr>
<tr>
<td>ModelANOVA</td>
<td>Model analysis of variance</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Estimated linear parameters</td>
</tr>
<tr>
<td>Spectral</td>
<td>Spectral analysis</td>
</tr>
<tr>
<td>StationaryPoint</td>
<td>Stationary point of response surface</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the RIDGE Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ridge</td>
<td>Ridge analysis for optimum response</td>
</tr>
</tbody>
</table>

**Table 10.67**  ODS Table Names Produced by the STDIZE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics</td>
<td>Location and scale measures</td>
<td>PSTAT</td>
</tr>
</tbody>
</table>
### Table 10.68  
ODS Table Names Produced by the STEPDISC Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCorr</td>
<td>Between-class correlations</td>
<td>BCORR</td>
</tr>
<tr>
<td>B Cov</td>
<td>Between-class covariances</td>
<td>BCOV</td>
</tr>
<tr>
<td>BSSCP</td>
<td>Between-class SSCP matrix</td>
<td>BSSCP</td>
</tr>
<tr>
<td>Counts</td>
<td>Number of observations,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>variables, classes, and DF</td>
<td></td>
</tr>
<tr>
<td>CovDF</td>
<td>DF for covariance matrices, not</td>
<td>any *COV option</td>
</tr>
<tr>
<td></td>
<td>printed</td>
<td></td>
</tr>
<tr>
<td>Levels</td>
<td>Class level information</td>
<td></td>
</tr>
<tr>
<td>Messages</td>
<td>Entry/removal messages</td>
<td></td>
</tr>
<tr>
<td>Multivariate</td>
<td>Multivariate statistics</td>
<td></td>
</tr>
<tr>
<td>PCorr</td>
<td>Pooled within-class correlations</td>
<td>PCORR</td>
</tr>
<tr>
<td>PCov</td>
<td>Pooled within-class covariances</td>
<td>PCOV</td>
</tr>
<tr>
<td>PSSCP</td>
<td>Pooled within-class SSCP matrix</td>
<td>PSSCP</td>
</tr>
<tr>
<td>PStdMeans</td>
<td>Pooled standardized class means</td>
<td>STDMEAN</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>Steps</td>
<td>Stepwise selection entry/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>removal</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>Stepwise selection summary</td>
<td></td>
</tr>
<tr>
<td>TCorr</td>
<td>Total-sample correlations</td>
<td>TCORR</td>
</tr>
<tr>
<td>T Cov</td>
<td>Total-sample covariances</td>
<td>TCOV</td>
</tr>
<tr>
<td>TSSCP</td>
<td>Total-sample SSCP matrix</td>
<td>TSSCP</td>
</tr>
<tr>
<td>TStdMeans</td>
<td>Total standardized class means</td>
<td>STDMEAN</td>
</tr>
<tr>
<td>Variables</td>
<td>Variable lists</td>
<td></td>
</tr>
<tr>
<td>WCov</td>
<td>Within-class covariances</td>
<td>WCOV</td>
</tr>
<tr>
<td>WSSCP</td>
<td>Within-class SSCP matrices</td>
<td>WSSCP</td>
</tr>
</tbody>
</table>
### Table 10.69 ODS Table Names Produced by the SURVEYMEANS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the CLASS Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClassVarInfo</td>
<td>Class level information</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the DOMAIN Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td>Statistics in domains</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the PROC Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>Data summary</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the RATIO Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td>Statistics for ratios</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the STRATA Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StrataInfo</td>
<td>Stratum information</td>
<td>LIST</td>
</tr>
</tbody>
</table>

### Table 10.70 ODS Table Names Produced by the SURVEYREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the CLASS Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClassVarInfo</td>
<td>Class level information</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the CLUSTER Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DesignSummary</td>
<td>Design summary</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the CONTRAST Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ContrastCoef</td>
<td>Coefficients of contrast</td>
<td>E</td>
</tr>
<tr>
<td>Contrasts</td>
<td>Analysis of contrasts</td>
<td></td>
</tr>
<tr>
<td><strong>ODS Tables Created by the ESTIMATE Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EstimateCoef</td>
<td>Coefficients of estimate</td>
<td>E</td>
</tr>
<tr>
<td>Estimates</td>
<td>Analysis of estimable functions</td>
<td></td>
</tr>
</tbody>
</table>
### ODS Tables Created by the MODEL Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>ANOVA for dependent variable</td>
<td>ANOVA</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of estimated</td>
<td>COVB</td>
</tr>
<tr>
<td></td>
<td>regression coefficients</td>
<td></td>
</tr>
<tr>
<td>DataSummary</td>
<td>Data summary</td>
<td></td>
</tr>
<tr>
<td>Effects</td>
<td>Tests of model effects</td>
<td></td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td></td>
</tr>
<tr>
<td>InvXPX</td>
<td>Inverse matrix of XX</td>
<td>INV</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Estimated regression coefficients</td>
<td></td>
</tr>
<tr>
<td>XPX</td>
<td>XX matrix</td>
<td>XPX</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the STRATA Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DesignSummary</td>
<td>Data summary</td>
<td></td>
</tr>
<tr>
<td>StrataInfo</td>
<td>Stratum information</td>
<td>LIST</td>
</tr>
</tbody>
</table>

### Table 10.71 ODS Table Names Produced by the SURVEYSELECT Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
</table>

### ODS Tables Created by the PROC Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
</table>

### Table 10.72 ODS Table Names Produced by the TPHREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
</table>

### ODS Tables Created by the CONTRAST Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
</table>

### ODS Tables Created by the MODEL Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>CensoredSummary</td>
<td>Summary of event and censored observations</td>
</tr>
<tr>
<td>ClassLevelInfo</td>
<td>CLASS variable levels and design variables</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
</tr>
<tr>
<td>CorrB</td>
<td>Estimated correlation matrix of parameter estimates</td>
</tr>
<tr>
<td>CovB</td>
<td>Estimated covariance matrix of parameter estimators</td>
</tr>
<tr>
<td>EffectsToEnter</td>
<td>Eligible effects for entry to model</td>
</tr>
<tr>
<td>EffectsToRemove</td>
<td>Eligible effects for removal from model</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Model fit statistics</td>
</tr>
<tr>
<td>GlobalScore</td>
<td>Global Chi-Square test</td>
</tr>
<tr>
<td>GlobalTests</td>
<td>Tests of the global null hypothesis</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
</tr>
<tr>
<td>LastGradient</td>
<td>Last evaluation of gradient</td>
</tr>
<tr>
<td>ModelBuildingSummary</td>
<td>Summary of model building</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Maximum likelihood estimates of model parameters</td>
</tr>
<tr>
<td>ResidualChiSq</td>
<td>Residual Chi-Square</td>
</tr>
<tr>
<td>Type3</td>
<td>Type 3 tests of effects</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the PROC Statement**

| ClassLevelFreq         | Frequency breakdown of CLASS variables           | SIMPLE (with CLASS variables) |
| ModelInfo              | Model information                                |                                 |
| SimpleStatistics       | Summary statistics for interval explanatory variables | SIMPLE              |

**ODS Tables Created by the TEST Statement**

| TestAverage            | Average effect for test                          | AVERAGE             |
| TestCoeff              | Coefficients for linear hypothesis               | E                   |
| TestPrint1             | L[cov(b)]L' and Lb-c                             | PRINT               |
| TestPrint2             | Ginv(L[cov(b)]L') and Ginv(L[cov(b)]L'( Lb-c))   | PRINT               |
| TestStmts              | Linear hypothesis test results                   |                     |
### ODS Tables Created by the WEIGHT Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassWgt</td>
<td>Weight breakdown of CLASS</td>
<td>SIMPLE (with CLASS</td>
</tr>
<tr>
<td></td>
<td>variables</td>
<td>variables)</td>
</tr>
</tbody>
</table>

### Table 10.73  ODS Table Names Produced by the TPSPLINE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCVFunction</td>
<td>GCV table</td>
<td>LOGNLAMBDA or LAMBDA</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the MODEL Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSummary</td>
<td>Data summary</td>
<td></td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Model fit statistics</td>
<td></td>
</tr>
<tr>
<td>FitSummary</td>
<td>Fit parameters and fit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>summary</td>
<td></td>
</tr>
</tbody>
</table>

### ODS Tables Created by the PROC Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>Equation</td>
<td>Linear dependency equation</td>
<td>less-than-full-rank model</td>
</tr>
<tr>
<td>Footnotes</td>
<td>Iteration history footnotes</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10.74  ODS Table Names Produced by the TRANSREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoxCox</td>
<td>Box-Cox transformation results</td>
<td>BOXCOX</td>
</tr>
<tr>
<td>SplineCoef</td>
<td>Spline coefficients</td>
<td>SPLINE or MSPLINE</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the MODEL Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>NObs</td>
<td>ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>ClassLevels</td>
<td>ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>ANOVA</td>
<td>ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>LiberalANOVA</td>
<td>ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>ConservANOVA</td>
<td>ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics like R-Square</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>LiberalFitStatistics</td>
<td>Fit statistics</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>ConservFitStatistics</td>
<td>Fit statistics</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>MVANOVA</td>
<td>Multivariate ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>LiberalMVANOVA</td>
<td>Multivariate ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>ConservMVANOVA</td>
<td>Multivariate ANOVA</td>
<td>TEST or SS2</td>
</tr>
<tr>
<td>Coef</td>
<td>Regression results</td>
<td>SS2</td>
</tr>
<tr>
<td>LiberalCoef</td>
<td>Regression results</td>
<td>SS2</td>
</tr>
<tr>
<td>ConservCoef</td>
<td>Regression results</td>
<td>SS2</td>
</tr>
<tr>
<td>MVCoef</td>
<td>Multivariate regression results</td>
<td>SS2</td>
</tr>
<tr>
<td>LiberalMVCoef</td>
<td>Multivariate regression results</td>
<td>SS2</td>
</tr>
<tr>
<td>ConservMVCoef</td>
<td>Multivariate regression results</td>
<td>SS2</td>
</tr>
<tr>
<td>Utilities</td>
<td>Conjoint analysis utilities</td>
<td>UTILITY</td>
</tr>
<tr>
<td>LiberalUtilities</td>
<td>Conjoint analysis utilities</td>
<td>UTILITY</td>
</tr>
<tr>
<td>ConservUtilities</td>
<td>Conjoint analysis utilities</td>
<td>UTILITY</td>
</tr>
<tr>
<td>Details</td>
<td>Model details</td>
<td>DETAIL</td>
</tr>
<tr>
<td>Univariate</td>
<td>Univariate iteration history</td>
<td>METHOD=UNIVARIATE</td>
</tr>
<tr>
<td>MORALS</td>
<td>MORALS iteration history</td>
<td>METHOD=MORALS</td>
</tr>
<tr>
<td>CANALS</td>
<td>CANALS iteration history</td>
<td>METHOD=CANALS</td>
</tr>
<tr>
<td>Redundancy</td>
<td>Redundancy iteration history</td>
<td>METHOD=REDUNDANCY</td>
</tr>
<tr>
<td>TestIterations</td>
<td>Hypothesis test iterations</td>
<td>SS2</td>
</tr>
</tbody>
</table>

**Table 10.75**  ODS Table Names Produced by the TREE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>Line-printer plot of the tree</td>
<td>LINEPRINTER</td>
</tr>
<tr>
<td>TreeListing</td>
<td>Line-printer listing of all nodes in the tree</td>
<td>LIST</td>
</tr>
</tbody>
</table>
### Table 10.76  ODS Table Names Produced by the TTEST Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics</td>
<td>Univariate summary statistics</td>
</tr>
<tr>
<td>TTests</td>
<td>$t$-tests</td>
</tr>
</tbody>
</table>

#### ODS Tables Created by the CLASS Statement

| Equality | Tests for equality of variance |

### Table 10.77  ODS Table Names Produced by the VARCLUS Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterQuality</td>
<td>Cluster quality</td>
<td></td>
</tr>
<tr>
<td>ClusterStructure</td>
<td>Cluster structure</td>
<td></td>
</tr>
<tr>
<td>ClusterSummary</td>
<td>Cluster summary</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>Corr</td>
<td>Correlations</td>
<td>CORR</td>
</tr>
<tr>
<td>DataOptSummary</td>
<td>Data and options summary table</td>
<td></td>
</tr>
<tr>
<td>InterClusterCorr</td>
<td>Inter-cluster correlations</td>
<td></td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>TRACE</td>
</tr>
<tr>
<td>RSquare</td>
<td>Cluster R-Square</td>
<td></td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>StdScoreCoeff</td>
<td>Standardized scoring coefficients</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10.78  ODS Table Names Produced by the VARCOMP Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassLevels</td>
<td>Class level information</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>Estimates</td>
<td>Variance component estimates (one variable)</td>
<td></td>
</tr>
</tbody>
</table>
### ODS Table Names and the SAS/ETS Procedures that Produce Them

The following table lists the output object table names which SAS/ETS procedures produce. You must license SAS/ETS software in order to produce these output objects. The table provides the name of each table, a description of what the table contains, and the option, if any, that creates the output object table. For more information about SAS/ETS procedures, see Table 10.79.

#### Table 10.79 ODS Table Names Produced by the ARIMA Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptive statistics</td>
<td>Descriptive statistics</td>
<td></td>
</tr>
<tr>
<td>Input descriptive statistics</td>
<td>Input descriptive statistics</td>
<td></td>
</tr>
<tr>
<td>Correlations graph</td>
<td>Correlations graph</td>
<td></td>
</tr>
<tr>
<td>Stationarity tests</td>
<td>Stationarity tests</td>
<td>STATIONARITY</td>
</tr>
<tr>
<td>Tentative order selections</td>
<td>Tentative order selections</td>
<td>MINIC, ESACF, or SCAN</td>
</tr>
<tr>
<td>Partial autocorrelations graph</td>
<td>Partial autocorrelations graph</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>IACFGraph</td>
<td>Inverse autocorrelations graph</td>
<td></td>
</tr>
<tr>
<td>ChiSqAuto</td>
<td>Chi-Square statistics table for autocorrelation</td>
<td></td>
</tr>
<tr>
<td>ChiSqCross</td>
<td>Chi-Square statistics table for cross-correlations</td>
<td>CROSSCORR=</td>
</tr>
<tr>
<td>MINIC</td>
<td>Minimum information criterion</td>
<td>MINIC</td>
</tr>
<tr>
<td>ESACF</td>
<td>Extended sample autocorrelation function</td>
<td>ESACF</td>
</tr>
<tr>
<td>ESACFPValues</td>
<td>ESACF probability values</td>
<td>ESACF</td>
</tr>
<tr>
<td>SCAN</td>
<td>Squared canonical correlation estimates</td>
<td>SCAN</td>
</tr>
<tr>
<td>SCANValues</td>
<td>SCAN Chi-Square[1] probability values</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the ESTIMATE Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td></td>
</tr>
<tr>
<td>ARPolynomial</td>
<td>Filter equations</td>
<td></td>
</tr>
<tr>
<td>MAPolynomial</td>
<td>Filter equations</td>
<td></td>
</tr>
<tr>
<td>NumPolynomial</td>
<td>Filter equations</td>
<td></td>
</tr>
<tr>
<td>DenPolynomial</td>
<td>Filter equations</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>ChiSqAuto</td>
<td>Chi-Square statistics table for autocorrelation</td>
<td></td>
</tr>
<tr>
<td>ChiSqCross</td>
<td>Chi-Square statistics table for cross-correlations</td>
<td></td>
</tr>
<tr>
<td>InitialAREstimates</td>
<td>Initial autoregressive parameter estimates</td>
<td></td>
</tr>
<tr>
<td>InitialMAEstimates</td>
<td>Initial moving average parameter estimates</td>
<td></td>
</tr>
<tr>
<td>PrelimEstimates</td>
<td>Preliminary estimation</td>
<td></td>
</tr>
<tr>
<td>IterHistory</td>
<td>Conditional least squares estimation</td>
<td>METHOD=CLS</td>
</tr>
<tr>
<td>OptSummary</td>
<td>ARIMA estimation optimization</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>ModelDescription</td>
<td>Model description</td>
<td></td>
</tr>
<tr>
<td>InputDescription</td>
<td>Input description</td>
<td></td>
</tr>
<tr>
<td>ObjectiveGrid</td>
<td>Objective function grid matrix</td>
<td>GRID</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlations of the estimates</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the OUTLIER Statement**
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutlierDetails</td>
<td>Detected outliers</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the FORECAST Statement**

<table>
<thead>
<tr>
<th>Forecasts</th>
<th>Fit statistics</th>
</tr>
</thead>
</table>

**Table 10.80** ODS Table Names Produced by the AUTOREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitSummary</td>
<td>Summary of regression</td>
<td></td>
</tr>
<tr>
<td>SummaryDepVarCen</td>
<td>Summary of regression (centered dependent variable)</td>
<td>CENTER</td>
</tr>
<tr>
<td>SummaryNoIntercept</td>
<td>Summary of regression (no intercept)</td>
<td>NOINT</td>
</tr>
<tr>
<td>YWIterSSE</td>
<td>Yule-Walker iteration sum of squared error</td>
<td>METHOD=ITYW</td>
</tr>
<tr>
<td>PreMSE</td>
<td>Preliminary MSEs</td>
<td>NLAG=</td>
</tr>
<tr>
<td>Dependent</td>
<td>Dependent variable</td>
<td></td>
</tr>
<tr>
<td>DependenceEquations</td>
<td>Linear dependence equation</td>
<td></td>
</tr>
<tr>
<td>ARCHTest</td>
<td>Q and LM tests for ARCH disturbances</td>
<td>ARCHTEST</td>
</tr>
<tr>
<td>ChowTest</td>
<td>Chow test and predictive chow test</td>
<td>CHOW= or PCHOW=</td>
</tr>
<tr>
<td>Godfrey</td>
<td>Godfrey’s serial correlation test</td>
<td>GODFREY or GODFREY=</td>
</tr>
<tr>
<td>PhilPerron</td>
<td>Phillips-Perron unit root test</td>
<td>STATIONARITY=, (PHILLIPS&lt;=()&gt;, (no regressor)</td>
</tr>
<tr>
<td>PhilOul</td>
<td>Phillips-Ouliaris cointegration test</td>
<td>STATIONARITY=, (PHILLIPS&lt;=()&gt;, (has regressor)</td>
</tr>
<tr>
<td>ResetTest</td>
<td>Ramsey’s RESET test</td>
<td>RESET</td>
</tr>
<tr>
<td>ARParameterEstimates</td>
<td>Estimates of autoregressive parameters</td>
<td>NLAG=</td>
</tr>
<tr>
<td>CorrGraph</td>
<td>Estimates of autocorrelations</td>
<td>NLAG=</td>
</tr>
<tr>
<td>BackStep</td>
<td>Backward elimination of autoregressive terms</td>
<td>BACKSTEP</td>
</tr>
<tr>
<td>ExpAutocorr</td>
<td>Expected autocorrelations</td>
<td>NLAG=</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimatesGivenAR</td>
<td>Parameter estimates assuming AR parameters are given</td>
<td>NLAG=</td>
</tr>
<tr>
<td>PartialAutoCorr</td>
<td>Partial autocorrelation</td>
<td>PARTIAL</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameter estimates</td>
<td>COVB</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation of parameter estimates</td>
<td>CORRB</td>
</tr>
<tr>
<td>CholeskyFactor</td>
<td>Cholesky root of gamma</td>
<td>ALL</td>
</tr>
<tr>
<td>Coefficients</td>
<td>Coefficients for first NLAG observations</td>
<td>COEF</td>
</tr>
<tr>
<td>GammaInverse</td>
<td>Gamma inverse</td>
<td>GINV</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status table</td>
<td></td>
</tr>
<tr>
<td>DWTestProb</td>
<td>Durbin-Watson statistics</td>
<td>DW=</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the RESTRICT Statement**

<table>
<thead>
<tr>
<th>Restrict</th>
<th>Restriction table</th>
</tr>
</thead>
</table>

**ODS Tables Created by the TEST Statement**

<table>
<thead>
<tr>
<th>FTest</th>
<th>F test</th>
</tr>
</thead>
<tbody>
<tr>
<td>WaldTest</td>
<td>Wald test TYPE=WALD</td>
</tr>
</tbody>
</table>

**Table 10.81  ODS Table Names Produced by the ENTROPY Procedure**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvCrit</td>
<td>Convergence criteria for estimation</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
</tr>
<tr>
<td>DatasetOptions</td>
<td>Data sets used</td>
</tr>
<tr>
<td>MinSummary</td>
<td>Number of parameters, estimation kind</td>
</tr>
<tr>
<td>ObsUsed</td>
<td>Observations read, used, and missing</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
</tr>
<tr>
<td>ResidSummary</td>
<td>Summary of the SSE, MSE for the equations</td>
</tr>
<tr>
<td>TestResults</td>
<td>Test statement table</td>
</tr>
</tbody>
</table>
### Table 10.82  ODS Table Names Produced by the LOAN Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ODS Tables Created by the PROC LOAN, FIXED, ARM, BALLOON, and BUYDOWN Statements</td>
<td></td>
</tr>
<tr>
<td>Repayment</td>
<td>Loan repayment schedule</td>
<td>SCHEDULE</td>
</tr>
<tr>
<td>LoanSummary</td>
<td>Loan summary</td>
<td></td>
</tr>
<tr>
<td>RateList</td>
<td>Rates and payments</td>
<td></td>
</tr>
<tr>
<td>PrepayList</td>
<td>Prepayments and periods</td>
<td>PREPAYMENTS=</td>
</tr>
<tr>
<td></td>
<td>ODS Tables Created by the BALLOON Statement</td>
<td></td>
</tr>
<tr>
<td>BalloonList</td>
<td>Balloon payments and periods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ODS Tables Created by the COMPARE Statement</td>
<td></td>
</tr>
<tr>
<td>Comparison</td>
<td>Loan comparison report</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10.83  ODS Table Names Produced by the MDC Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ODS Tables Created by the MODEL Statement</td>
<td></td>
</tr>
<tr>
<td>FitSummary</td>
<td>Summary of nonlinear estimation</td>
<td></td>
</tr>
<tr>
<td>ResponseProfile</td>
<td>Response profile</td>
<td></td>
</tr>
<tr>
<td>GoodnessOfFit</td>
<td>Pseudo-R² measures</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>LinConSol</td>
<td>Linearly independent active linear constraints</td>
<td></td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameter estimates</td>
<td>COVB</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation of parameter estimates</td>
<td>CORRB</td>
</tr>
</tbody>
</table>
### Table 10.84 ODS Table Names Produced by the MODEL Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the FIT Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AugGMMCovariance</td>
<td>Cross products matrix</td>
<td>GMM</td>
</tr>
<tr>
<td>ChowTest</td>
<td>Structural change test</td>
<td>CHOW=</td>
</tr>
<tr>
<td>CollinDiagnostics</td>
<td>Collinearity diagnostics</td>
<td></td>
</tr>
<tr>
<td>ConfInterval</td>
<td>Profile likelihood confidence intervals</td>
<td>PRL=</td>
</tr>
<tr>
<td>ConvCrit</td>
<td>Convergence criteria for estimation</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlations of parameters</td>
<td>COVB or CORRB</td>
</tr>
<tr>
<td>CorrResiduals</td>
<td>Correlations of residuals</td>
<td>CORRS or COVS</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameters</td>
<td>COVB or CORRB</td>
</tr>
<tr>
<td>CovResiduals</td>
<td>Covariance of residuals</td>
<td>CORRS or COVS</td>
</tr>
<tr>
<td>Crossproducts</td>
<td>Cross products matrix</td>
<td>ITALL or ITPRINT</td>
</tr>
<tr>
<td>DatasetOptions</td>
<td>Data sets used</td>
<td></td>
</tr>
<tr>
<td>DetResidCov</td>
<td>Determinant of the residuals</td>
<td>DETAILS</td>
</tr>
<tr>
<td>DWTest</td>
<td>Durbin-Watson test</td>
<td>DW=</td>
</tr>
<tr>
<td>Equations</td>
<td>List of equations to estimate</td>
<td></td>
</tr>
<tr>
<td>EstSummaryMiss</td>
<td>Model summary statistics for PAIRWISE</td>
<td>MISSING=</td>
</tr>
<tr>
<td>EstSummaryStats</td>
<td>Objective, objective * N</td>
<td></td>
</tr>
<tr>
<td>GMMCovariance</td>
<td>Cross products matrix</td>
<td>GMM</td>
</tr>
<tr>
<td>Godfrey</td>
<td>Godfrey's serial correlation test</td>
<td>GF=</td>
</tr>
<tr>
<td>HausmanTest</td>
<td>Hausman's test table</td>
<td>HAUSMAN</td>
</tr>
<tr>
<td>HeteroTest</td>
<td>Heteroscedasticity test tables</td>
<td>BREUSCH or PAGEN</td>
</tr>
<tr>
<td>InvXPXMat</td>
<td>$XX$ inverse for system</td>
<td>$I$</td>
</tr>
<tr>
<td>IterInfo</td>
<td>Iteration printing</td>
<td>ITALL or ITPRINT</td>
</tr>
<tr>
<td>LagLength</td>
<td>Model lag length</td>
<td></td>
</tr>
<tr>
<td>MinSummary</td>
<td>Number of parameters, estimation kind</td>
<td></td>
</tr>
<tr>
<td>MissingValues</td>
<td>Missing values generated by the program</td>
<td></td>
</tr>
<tr>
<td>ModSummary</td>
<td>List of all categorized values</td>
<td></td>
</tr>
<tr>
<td>ModVars</td>
<td>List of model variables and parameters</td>
<td></td>
</tr>
<tr>
<td>NormalityTest</td>
<td>Normality test table</td>
<td>NORMAL</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ObsSummary</td>
<td>Identifies observations with errors</td>
<td></td>
</tr>
<tr>
<td>ObsUsed</td>
<td>Observations read, used, and missing</td>
<td>default</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>ParmChange</td>
<td>Parameter change vector</td>
<td></td>
</tr>
<tr>
<td>ResidSummary</td>
<td>Summary of the SSE, MSE for the equations</td>
<td></td>
</tr>
<tr>
<td>SizeInfo</td>
<td>Storage requirement for estimation</td>
<td>DETAILS</td>
</tr>
<tr>
<td>TermEstimates</td>
<td>Nonlinear OLS and ITOLS estimates</td>
<td>OLS or ITOLS</td>
</tr>
<tr>
<td>TestResults</td>
<td>Test statement table</td>
<td></td>
</tr>
<tr>
<td>WgtVar</td>
<td>The name of the weight variable</td>
<td></td>
</tr>
<tr>
<td>XPXMat</td>
<td>XX for system</td>
<td>XPX</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the SOLVE Statement**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatasetOptions</td>
<td>Data sets used</td>
<td></td>
</tr>
<tr>
<td>DescriptiveStatistics</td>
<td>Descriptive statistics</td>
<td>STATS</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics for simulation</td>
<td>STATS</td>
</tr>
<tr>
<td>LagLength</td>
<td>Model lag length</td>
<td></td>
</tr>
<tr>
<td>ModSummary</td>
<td>List of all categorized variables</td>
<td></td>
</tr>
<tr>
<td>ObsSummary</td>
<td>Simulation trace output</td>
<td>SOLVEPRINT</td>
</tr>
<tr>
<td>ObsUsed</td>
<td>Observations resa, used, and missing</td>
<td></td>
</tr>
<tr>
<td>SimulationSummary</td>
<td>Number of variables solved for</td>
<td></td>
</tr>
<tr>
<td>SolutionVarList</td>
<td>Solution variable lists</td>
<td></td>
</tr>
<tr>
<td>TheilRelStats</td>
<td>Theil relative change error statistics</td>
<td>THEIL</td>
</tr>
<tr>
<td>TheilStats</td>
<td>Theil forecast error statistics</td>
<td>THEIL</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the FIT and SOLVE Statements**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdjacencyMatrix</td>
<td>Adjacency graph</td>
<td>GRAPH</td>
</tr>
<tr>
<td>BlockAnalysis</td>
<td>Block analysis</td>
<td>BLOCK</td>
</tr>
<tr>
<td>CodeDependency</td>
<td>Variable cross reference</td>
<td>LISTDEP</td>
</tr>
<tr>
<td>CodeList</td>
<td>List of programs statements</td>
<td>LISTCODE</td>
</tr>
<tr>
<td>CrossReference</td>
<td>Cross reference listing for program</td>
<td></td>
</tr>
</tbody>
</table>
### Table Name | Description | Option
---|---|---
DepStructure | Dependency structure for the system | BLOCK
DerList | Derivative variables | LISTDER
InterIntg | Integration iteration output | INTGPRINT
MemUsage | Memory usage statistics | MEMORYUSE
ParmReadIn | Parameter estimates read in | ESTDATA=
ProgList | List of compiled program data | 
RangeInfo | RANGE statement specification | 
SortAdjacencyMatrix | Sorted adjacency graph | GRAPH
TransitiveClosure | Transitive closure graph | GRAPH

### Table 10.85  ODS Table Names Produced by the PDLREG Procedure

| Table Name | Description | Option |
---|---|---
ARParameterEstimates | Estimates of autoregressive parameters | NLAG=
CholeskyFactor | Cholesky root of gamma | 
Coefficients | Coefficients for first NLAG observations | NLAG=
ConvergenceStatus | Convergence status table | 
CorrB | Correlation of parameter estimates | CORRB
CorrGraph | Estimates of autocorrelations | NLAG=
CovB | Covariance of parameter estimates | COVB
DependenceEquations | Linear dependence equation | 
Dependent | Dependent variable | 
DWTest | Durbin-Watson statistics | DW=
ExpAutocorr | Expected autocorrelations | NLAG=
FitSummary | Summary of regression | 
GammaInverse | Gamma inverse | 
IterHistory | Iteration history | ITPRINT
LagDist | Lag distribution | ALL
ParameterEstimates | Parameter estimates | 

**ODS Tables Created by the MODEL Statement**

| Table Name | Description | Option |
---|---|---
ARParameterEstimates | Estimates of autoregressive parameters | NLAG=
CholeskyFactor | Cholesky root of gamma | 
Coefficients | Coefficients for first NLAG observations | NLAG=
ConvergenceStatus | Convergence status table | 
CorrB | Correlation of parameter estimates | CORRB
CorrGraph | Estimates of autocorrelations | NLAG=
CovB | Covariance of parameter estimates | COVB
DependenceEquations | Linear dependence equation | 
Dependent | Dependent variable | 
DWTest | Durbin-Watson statistics | DW=
ExpAutocorr | Expected autocorrelations | NLAG=
FitSummary | Summary of regression | 
GammaInverse | Gamma inverse | 
IterHistory | Iteration history | ITPRINT
LagDist | Lag distribution | ALL
ParameterEstimates | Parameter estimates | 

**TEMPLATE Procedure: Creating Tabular Output**

**ODS Output Object Table Names**
### Table 10.86  ODS Table Names Produced by the SIMLIN Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endogenous</td>
<td>Structural coefficients for endogenous variables</td>
<td></td>
</tr>
<tr>
<td>LaggedEndogenous</td>
<td>Structural coefficients for lagged endogenous variables</td>
<td></td>
</tr>
<tr>
<td>Exogenous Structural</td>
<td>Coefficients for exogenous variables</td>
<td></td>
</tr>
<tr>
<td>InverseCoeff</td>
<td>Inverse coefficient matrix for endogenous variables</td>
<td></td>
</tr>
<tr>
<td>RedFormLagEndo</td>
<td>Reduced form for lagged endogenous variables</td>
<td></td>
</tr>
<tr>
<td>RedFormExog</td>
<td>Reduced form for exogenous variables</td>
<td></td>
</tr>
<tr>
<td>InterimMult</td>
<td>Interim multipliers</td>
<td>INTERIM=option</td>
</tr>
<tr>
<td>TotalMult</td>
<td>Total multipliers</td>
<td>TOTAL=option</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10.87  ODS Table Names Produced by the SPECTRA Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>WhiteNoiseTest</td>
<td>White noise test</td>
<td>WHITETEST</td>
</tr>
</tbody>
</table>
### Table 10.88  ODS Table Names Produced by the STATESPACE Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>NObs</td>
<td>Number of observations</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>Simple summary statistics table</td>
<td></td>
</tr>
<tr>
<td>InfoCriterion</td>
<td>Information criterion table</td>
<td></td>
</tr>
<tr>
<td>CovLags</td>
<td>Covariance matrices of input series</td>
<td>PRINTOUT=LONG</td>
</tr>
<tr>
<td>CorrLags</td>
<td>Correlation matrices of input series</td>
<td>PRINTOUT=LONG</td>
</tr>
<tr>
<td>PartialAR</td>
<td>Partial autoregressive matrices</td>
<td>PRINTOUT=LONG</td>
</tr>
<tr>
<td>YWEstimates</td>
<td>Yule-Walker estimates for minimum AIC</td>
<td></td>
</tr>
<tr>
<td>CovResiduals</td>
<td>Covariance of residuals</td>
<td>PRINTOUT=LONG</td>
</tr>
<tr>
<td>CorrResiduals</td>
<td>Residual correlations from AR models</td>
<td>PRINTOUT=LONG</td>
</tr>
<tr>
<td>StateVector</td>
<td>State vector table</td>
<td></td>
</tr>
<tr>
<td>CorrGraph</td>
<td>Schematic representation of correlations</td>
<td></td>
</tr>
<tr>
<td>TransitionMatrix</td>
<td>Transition matrix</td>
<td></td>
</tr>
<tr>
<td>InputMatrix</td>
<td>Input matrix</td>
<td></td>
</tr>
<tr>
<td>VarInnov</td>
<td>Variance matrix for the innovation</td>
<td></td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameter estimates</td>
<td>COVB</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation of parameter estimates</td>
<td>COVB</td>
</tr>
<tr>
<td>CanCorr</td>
<td>Canonical correlation analysis</td>
<td>CANCORR</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iterative fitting table</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates table</td>
<td></td>
</tr>
<tr>
<td>Forecasts</td>
<td>Forecasts table</td>
<td>PRINT</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status table</td>
<td></td>
</tr>
</tbody>
</table>
### Table 10.89  ODS Table Names Produced by the SYSLIN Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Summary of the SSE, MSE for the equations</td>
<td></td>
</tr>
<tr>
<td>AugXPXMat</td>
<td>Model crossproducts</td>
<td>XPX</td>
</tr>
<tr>
<td>AutoCorrStat</td>
<td>Autocorrelation statistics</td>
<td></td>
</tr>
<tr>
<td>ConvCrit</td>
<td>Convergence criteria for estimation</td>
<td></td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlations of parameters</td>
<td>CORRB</td>
</tr>
<tr>
<td>CorrResiduals</td>
<td>Correlations of residuals</td>
<td>CORRS</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameters</td>
<td>COVB</td>
</tr>
<tr>
<td>CovResiduals</td>
<td>Covariance of residuals</td>
<td>COVS</td>
</tr>
<tr>
<td>Endomat</td>
<td>Endogenous variables</td>
<td></td>
</tr>
<tr>
<td>Equations</td>
<td>List of equations to estimates</td>
<td></td>
</tr>
<tr>
<td>ExogMat</td>
<td>Exogenous variables</td>
<td></td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Statistics of fit</td>
<td></td>
</tr>
<tr>
<td>InvCorrResiduals</td>
<td>Inverse correlations of residuals</td>
<td>CORRS</td>
</tr>
<tr>
<td>InvCovResiduals</td>
<td>Inverse covariance of residuals</td>
<td>COVS</td>
</tr>
<tr>
<td>InvEndoMat</td>
<td>Inverse endogenous variables</td>
<td></td>
</tr>
<tr>
<td>InvXPX</td>
<td>XX inverse for system</td>
<td>I</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration printing</td>
<td>ITALL or ITPRINT</td>
</tr>
<tr>
<td>MissingValues</td>
<td>Missing values generated by the program</td>
<td></td>
</tr>
<tr>
<td>ModelVars</td>
<td>Name and label for the model</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>RedMat</td>
<td>Reduced form</td>
<td>REDUCED</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Descriptive statistics</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>SSCP</td>
<td>Model crossproducts</td>
<td></td>
</tr>
<tr>
<td>TestResults</td>
<td>Test for overidentifying restrictions</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>Weighted model statistics</td>
<td></td>
</tr>
<tr>
<td>YPY</td>
<td>YY matrices</td>
<td>USSCP2</td>
</tr>
</tbody>
</table>
### Table 10.90  ODS Table Names Produced by the TSCSREG Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelDescription</td>
<td>Model description</td>
<td></td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td></td>
</tr>
<tr>
<td>FixedEffectsTest</td>
<td>F test for no fixed tests</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td></td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameter estimates</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlations of parameter estimates</td>
<td></td>
</tr>
<tr>
<td>VarianceComponents</td>
<td>Variance component estimates</td>
<td></td>
</tr>
<tr>
<td>RandomEffectsTest</td>
<td>Hausman test for random effects</td>
<td></td>
</tr>
<tr>
<td>AR1Estimates</td>
<td>First order autoregressive parameter estimates</td>
<td></td>
</tr>
<tr>
<td>EstimatedPhiMatrix</td>
<td>Estimated phi matrix</td>
<td>PARKS</td>
</tr>
<tr>
<td>EstimatedAutocovariances</td>
<td>Estimates of autocovariances</td>
<td>PARKS</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the TEST Statement

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestResults</td>
<td>Test results</td>
</tr>
</tbody>
</table>

### Table 10.91  ODS Table Names Produced by the TIMESERIES Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeasonalDecomposition</td>
<td>Seasonal decomposition</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the PRINT=DECOMP Option

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeasonalDecomposition</td>
<td>Seasonal decomposition</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the PRINT=DESCSTATS Option

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DescStats</td>
<td>Descriptive statistics</td>
</tr>
</tbody>
</table>

### ODS Tables Created by the PRINT=SEASONS Option

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeasonStatistics</td>
<td>Season statistics</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>ODS Tables Created by the PRINT=TRENDS Option</td>
<td></td>
</tr>
<tr>
<td>TrendStatistics</td>
<td>Trend statistics</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Table 10.92</td>
<td>ODS Table Names Produced by the VARMAX Procedure</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ODS Tables Created by the MODEL Statement</td>
<td></td>
</tr>
<tr>
<td>AccumImpulse</td>
<td>Accumulated impulse response matrices</td>
</tr>
<tr>
<td>AccumImpulsX</td>
<td>Accumulated transfer function matrices</td>
</tr>
<tr>
<td>Alpha</td>
<td>(\alpha) coefficients</td>
</tr>
<tr>
<td>AlphaInECM</td>
<td>(\alpha) coefficients</td>
</tr>
<tr>
<td>AlphaOnDrift</td>
<td>(\alpha) coefficients on restriction of a deterministic term</td>
</tr>
<tr>
<td>AlphaBetaInECM</td>
<td>(\pi=\alpha\beta) coefficients</td>
</tr>
<tr>
<td>ArchCoef</td>
<td>ARCH coefficients</td>
</tr>
<tr>
<td>ARCoef</td>
<td>AR coefficients</td>
</tr>
<tr>
<td>ARRoots</td>
<td>Roots of AR characteristic polynomial</td>
</tr>
<tr>
<td>Beta</td>
<td>(\beta) coefficients</td>
</tr>
<tr>
<td>BetaInECM</td>
<td>(\beta) coefficients</td>
</tr>
<tr>
<td>BetaOnDrift</td>
<td>(\beta) coefficients on restriction of a deterministic term</td>
</tr>
<tr>
<td>Constant</td>
<td>Constant estimates</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlations of parameter estimates</td>
</tr>
<tr>
<td>CorrResiduals</td>
<td>Cross-correlations of residuals</td>
</tr>
<tr>
<td>CorrResidualsGraph</td>
<td>Schematic representation of residual cross-correlations</td>
</tr>
<tr>
<td>CorrGraph</td>
<td>Schematic representation of sample cross-correlations</td>
</tr>
<tr>
<td>CorrXLags</td>
<td>Cross-correlation matrices of independent series</td>
</tr>
<tr>
<td>CorrYLags</td>
<td>Cross-correlation matrices of dependent series</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance of parameter estimates</td>
</tr>
<tr>
<td>CovInnov</td>
<td>Covariance matrix for the innovation</td>
</tr>
<tr>
<td>CovPredError</td>
<td>Covariance matrices of the prediction error</td>
</tr>
<tr>
<td>CovResiduals</td>
<td>Cross-covariance matrices of residuals</td>
</tr>
<tr>
<td>CovXLags</td>
<td>Cross-covariance matrices of independent series</td>
</tr>
<tr>
<td>CovYLags</td>
<td>Cross-correlations matrices of dependent series</td>
</tr>
<tr>
<td>DecompCovPredError</td>
<td>Decomposition of the prediction error covariance</td>
</tr>
<tr>
<td>DFTest</td>
<td>Dickey-Fuller tests</td>
</tr>
<tr>
<td>DriftHypo</td>
<td>Hypothesis of different deterministic terms in cointegration rank test</td>
</tr>
<tr>
<td>DrifyHypoTest</td>
<td>Test hypothesis of different deterministic terms in cointegration rank test</td>
</tr>
<tr>
<td>EigenvalueI2</td>
<td>Eigenvalues in integrated order 2</td>
</tr>
<tr>
<td>Eta</td>
<td>$\eta$ coefficients</td>
</tr>
<tr>
<td>GARCHParameterEstimates</td>
<td>GARCH parameter estimates table</td>
</tr>
<tr>
<td>GARCHParameterGraph</td>
<td>Schematic representation of the garch parameters</td>
</tr>
<tr>
<td>GARCHRoots</td>
<td>Roots of GARCH characteristic polynomial</td>
</tr>
<tr>
<td>GARCHCoef</td>
<td>GARCH coefficients</td>
</tr>
<tr>
<td>GARCHConstant</td>
<td>GARCH constant estimates</td>
</tr>
<tr>
<td>InfiniteARRepresent</td>
<td>Infinite order AR representation</td>
</tr>
<tr>
<td>InfoCriterion</td>
<td>Information criterion</td>
</tr>
<tr>
<td>LinearTrend</td>
<td>Linear trend estimates</td>
</tr>
<tr>
<td>MACoef</td>
<td>MA coefficients</td>
</tr>
<tr>
<td>MARoots</td>
<td>Roots of MA characteristic polynomial</td>
</tr>
<tr>
<td>MaxTest</td>
<td>Cointegration rank test using the maximum eigenvalue</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MaxTestOnDrift</td>
<td>Cointegration rank test using the maximum eigenvalue on restriction of a deterministic term</td>
</tr>
<tr>
<td>ModelType</td>
<td>Type of model</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations</td>
</tr>
<tr>
<td>OrthoImpulse</td>
<td>Orthogonalized impulse response matrices</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates table</td>
</tr>
<tr>
<td>ParameterGraph</td>
<td>Schematic representation of the parameters</td>
</tr>
<tr>
<td>PartialAR</td>
<td>Partial autoregression matrices</td>
</tr>
<tr>
<td>PartialARGraph</td>
<td>Schematic representation of partial autoregression</td>
</tr>
<tr>
<td>PartialCanCorr</td>
<td>Partial canonical correlation analysis</td>
</tr>
<tr>
<td>PartialCorr</td>
<td>Partial cross-correlation matrices</td>
</tr>
<tr>
<td>PartialCorrGraph</td>
<td>Schematic representation of partial cross correlations</td>
</tr>
<tr>
<td>PortmanteauTest</td>
<td>Chi-Square test table for residual cross-correlations</td>
</tr>
<tr>
<td>ProportionDecomp</td>
<td>Proportions of prediction error covariance decomposition</td>
</tr>
<tr>
<td>RankTestI2</td>
<td>Cointegration rank test in integrated order 2</td>
</tr>
<tr>
<td>QuadTrend</td>
<td>Quadratic trend estimates</td>
</tr>
<tr>
<td>SConstant</td>
<td>Seasonal constant estimates</td>
</tr>
<tr>
<td>SimpleImpulse</td>
<td>Impulse response matrices</td>
</tr>
<tr>
<td>SimpleImpulseX</td>
<td>Impulse response matrices in transfer function</td>
</tr>
<tr>
<td>Summary</td>
<td>Simple summary statistics</td>
</tr>
<tr>
<td>SWTTest</td>
<td>Common trends test</td>
</tr>
<tr>
<td>TentativeOrders</td>
<td>Tentative order selection</td>
</tr>
<tr>
<td>TraceTest</td>
<td>Cointegration rank test using the trace</td>
</tr>
<tr>
<td>TraceTestOnDrift</td>
<td>Cointegration rank test using the trace on restriction of a deterministic term</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>UnivarDiagnostAR</td>
<td>Check the AR disturbance for the residuals</td>
</tr>
<tr>
<td>UnivarDiagnostCheck</td>
<td>Univariate model diagnostic checks</td>
</tr>
<tr>
<td>UnivarDiagnostTest</td>
<td>Check the ARCH disturbance and normality for the residuals</td>
</tr>
<tr>
<td>Xi</td>
<td>ξ coefficient matrix</td>
</tr>
<tr>
<td>XLagCoef</td>
<td>Dependent coefficients</td>
</tr>
<tr>
<td>YWEstimates</td>
<td>Yule-Walker estimates</td>
</tr>
<tr>
<td>ByVariable</td>
<td>Prints by variable</td>
</tr>
</tbody>
</table>

**ODS Tables Created by the COINTEG Statement**

- AlphaInECM $\alpha$ coefficients
- AlphaBetaInECM $\pi = \alpha \beta$ coefficients
- BetaInECM $\beta$ coefficients
- AlphaOnTest $\alpha$ coefficients under restriction $H=$ or $J=$
- BetaOnTest $\beta$ coefficients under restriction $H=$ or $J=$
- RestrictMatrix Restriction matrix for $\alpha$ or $\beta$ $H=$ or $J=$
- RestrictTest Hypothesis testing of $\alpha$ or $\beta$ $H=$ or $J=$
- WeakExogeneity Testing weak exogeneity of each dependent variable with respect to beta EXOGENEITY

**ODS Tables Created by the CASUAL Statement**

- Causality Granger-Causality test

**ODS Tables Created by the RESTRICT Statement**

- Restrict Restriction table

**ODS Tables Created by the TEST Statement**

- Test Wald test

**ODS Tables Created by the OUTPUT Statement**

- Forecasts Forecasts table w/o NOPRINT
Table 10.93  ODS Table Names Produced by the X11 Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ODS Tables Created by the MONTHLY and QUARTERLY Statements</strong></td>
<td></td>
</tr>
<tr>
<td>Preface</td>
<td>X11 seasonal adjustment program information giving credits, dates, etc.</td>
<td>Always printed unless NOPRINT</td>
</tr>
<tr>
<td>A1</td>
<td>Table A1: OriginalSeries</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Table A2: Prior monthly</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>Table A3: Original series adjusted for prior monthly factors</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>Table A4: Prior trading day adjustment factors with and without length of month adjustments</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Table A5: Original series adjusted for priors</td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td>Table B16: Original series or original series adjusted for priors</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>Table B2: Trend cycle — centered nn-term moving average</td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>Table B3: Unmodified SI ratios</td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>Table B4: Replacement values for extreme SI ratios</td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td>Table B5: Seasonal factors</td>
<td></td>
</tr>
<tr>
<td>B6</td>
<td>Table B6: Seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>B7</td>
<td>Table B7: Trend cycle — Henderson curve</td>
<td></td>
</tr>
<tr>
<td>B8</td>
<td>Table B8: Unmodified SI ratios</td>
<td></td>
</tr>
<tr>
<td>B9</td>
<td>Table B9: Replacement values for extreme SI ratios</td>
<td></td>
</tr>
<tr>
<td>B10</td>
<td>Table B10: Seasonal factors</td>
<td></td>
</tr>
<tr>
<td>B11</td>
<td>Table B11: Seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>B13</td>
<td>Table B13: Irregular series</td>
<td></td>
</tr>
<tr>
<td>B15</td>
<td>Table B15: Preliminary trading day regression</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>B16</td>
<td>Table B16: Trading day adjustment factors derived from regression</td>
<td></td>
</tr>
<tr>
<td>B17</td>
<td>Table B17: Preliminary weights for irregular components</td>
<td></td>
</tr>
<tr>
<td>B18</td>
<td>Table B18: Trading day adjustment factors from combined weights</td>
<td></td>
</tr>
<tr>
<td>B19</td>
<td>Table B19: Original series adjusted for preliminary combined TD weights</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>Table C1: Original series adjusted for preliminary weights</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>Table C2: Trend cycle — centered nn-term moving average</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Table C4: Modified SI ratios</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>Table C5: Seasonal factors</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>Table C6: Seasonally adjusted factors</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>Table C7: Trend cycle — Henderson curve</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>Table C9: Modified CI ratios</td>
<td></td>
</tr>
<tr>
<td>C10</td>
<td>Table C10: Seasonal factors</td>
<td></td>
</tr>
<tr>
<td>C11</td>
<td>Table C11: Seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>C13</td>
<td>Table C13: Irregular series</td>
<td></td>
</tr>
<tr>
<td>C15</td>
<td>Table C15: Final trading day regression</td>
<td></td>
</tr>
<tr>
<td>C16</td>
<td>Table C16: Trading day adjustment factors derived from regression</td>
<td></td>
</tr>
<tr>
<td>C17</td>
<td>Table C17: Final weights for irregular component</td>
<td></td>
</tr>
<tr>
<td>C18</td>
<td>Table C18: Trading day adjustment factors from combined weights</td>
<td></td>
</tr>
<tr>
<td>C19</td>
<td>Table C19: Original series adjusted for final combined TD weights</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>D1</td>
<td>Table D1: Original series adjusted for final weights on nn-term moving average</td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>Table D4: Modified SI ratios</td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>Table D5: Seasonal factors</td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>Table D6: Seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>Table D7: Trend cycle — Henderson curve</td>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>Table D8: Final unmodified SI ratios</td>
<td></td>
</tr>
<tr>
<td>D10</td>
<td>Table D10: Final season factors</td>
<td></td>
</tr>
<tr>
<td>D11</td>
<td>Table D11: Final seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>D12</td>
<td>Table D12: Final trend cycle — Henderson curve</td>
<td></td>
</tr>
<tr>
<td>D13</td>
<td>Table D13: Final irregular series</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>Table E1: Original series modified for extremes</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>Table E2: Modified seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>Table E3: Modified irregular series</td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>Table E5: Month-to-month changes in original series</td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>Table E6: Month-to-month changes in final seasonally adjusted series</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>Table F1: MCD moving average</td>
<td></td>
</tr>
<tr>
<td>A13</td>
<td>Table A13: ARIMA forecasts</td>
<td>ARIMA statement</td>
</tr>
<tr>
<td>A14</td>
<td>Table A14: ARIMA backcasts</td>
<td>ARIMA statement</td>
</tr>
<tr>
<td>A15</td>
<td>Table A15: ARIMA extrapolation</td>
<td>ARIMA statement</td>
</tr>
<tr>
<td>B14</td>
<td>Table B14: Irregular values excluded from trading day regression</td>
<td></td>
</tr>
<tr>
<td>C14</td>
<td>Table C14: Irregular values excluded from trading day regression</td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>Table D9: Final replacement values</td>
<td></td>
</tr>
<tr>
<td>PriorDailyWgts</td>
<td>Adjusted prior daily weights</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>TDR_0</td>
<td>Final/preliminary trading day regression, part 1</td>
<td>MONTHLY only,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDREGR=ADJUST, TEST</td>
</tr>
<tr>
<td>TDR_1</td>
<td>Final/preliminary trading day regression, part 2</td>
<td>MONTHLY only,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDREGR=ADJUST, TEST</td>
</tr>
<tr>
<td>StandErrors</td>
<td>Standard errors of trading day adjustment factors</td>
<td>MONTHLY only,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDREGR=ADJUST, TEST</td>
</tr>
<tr>
<td>D9A</td>
<td>Year-to-year change in irregular and seasonal components and moving seasonal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ratio</td>
<td></td>
</tr>
<tr>
<td>StableSeasTest</td>
<td>Stable seasonality test</td>
<td>MONTHLY only</td>
</tr>
<tr>
<td>StableSeasFTest</td>
<td>Stable seasonality test</td>
<td>MONTHLY only</td>
</tr>
<tr>
<td>f2a</td>
<td>F2 summary measures, part 1</td>
<td></td>
</tr>
<tr>
<td>f2b</td>
<td>F2 summary measures, part 2</td>
<td></td>
</tr>
<tr>
<td>f2c</td>
<td>F2 summary measures, part 3</td>
<td></td>
</tr>
<tr>
<td>f2d</td>
<td>I/C ratio for monthly/quarterly span</td>
<td></td>
</tr>
<tr>
<td>f2f</td>
<td>Average percent change with regard to sign and standard over span</td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>Differences or ratios of annual totals, original and adjusted series</td>
<td></td>
</tr>
<tr>
<td>ChartG1</td>
<td>Chart G1</td>
<td></td>
</tr>
<tr>
<td>ChartG2</td>
<td>Chart G2</td>
<td></td>
</tr>
</tbody>
</table>

**ODS Tables Created by the ARIMA Statement**

- **CriteriaSummary**: Criteria summary ARIMA statement
- **ConvergeSummary**: Convergence summary
- **ArimaEst**: ARIMA estimation results, part 1
- **ArimaEst2**: ARIMA estimation results, part 2
- **Model_Summary**: Model summary
- **Ljung_BoxQ**: Table of Ljung-Box Q statistics
- **A13**: Table A13: ARIMA forecasts
- **A14**: Table A14: ARIMA backcasts
- **A15**: Table A15: ARIMA extrapolation

**ODS Tables Created by the SSPAN Statement**
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPR0A_1</td>
<td>S 0.A sliding spans analysis, number, and length of spans</td>
</tr>
<tr>
<td>SpanDates</td>
<td>S 0.A sliding spans analysis: dates of spans</td>
</tr>
<tr>
<td>SPR0B</td>
<td>S 0.B summary of F-tests for stable and moving seasonality</td>
</tr>
<tr>
<td>SPR1_1</td>
<td>S 1.A range analysis of seasonal factors</td>
</tr>
<tr>
<td>SPR1_b</td>
<td>S 1.B summary of range measures</td>
</tr>
<tr>
<td>SPRXA</td>
<td>2XA.1 breakdown of differences by month or quarter</td>
</tr>
<tr>
<td>SPRXB_2</td>
<td>S X.B histogram of flagged observation</td>
</tr>
<tr>
<td>SPRXA_2</td>
<td>S X.A.2 breakdowns of differences by year</td>
</tr>
<tr>
<td>MpdStats</td>
<td>S X.C: Statistics for maximum percentage differences</td>
</tr>
<tr>
<td>S_X_A_3</td>
<td>S 2.X.3 breakdown summary of flagged observation</td>
</tr>
<tr>
<td>SPR7_X</td>
<td>S 7.X sliding spans analysis PRINTALL</td>
</tr>
</tbody>
</table>

### Table 10.94  ODS Table Names Produced by the X12 Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Original series</td>
</tr>
<tr>
<td>A2</td>
<td>Prior-adjustment factors</td>
</tr>
<tr>
<td>RegParameterEstimates</td>
<td>Regression model parameter estimates</td>
</tr>
<tr>
<td>ACF</td>
<td>Autocorrelation factors</td>
</tr>
<tr>
<td>PACF</td>
<td>Partial autorrelation factors</td>
</tr>
<tr>
<td>ARMAIterationTolerances</td>
<td>Exact ARMA likelihood estimation iteration tolerances</td>
</tr>
<tr>
<td>IterHistory</td>
<td>ARMA iteration history</td>
</tr>
<tr>
<td>ARMAIterationSummary</td>
<td>Exact ARMA likelihood estimation iteration summary</td>
</tr>
<tr>
<td>RegressorGroupChiSq</td>
<td>Chi-Squared tests for groups of regressors</td>
</tr>
<tr>
<td>ARMAParameterEstimates</td>
<td>Exact ARMA maximum likelihood estimation</td>
</tr>
<tr>
<td>AvgFestErr</td>
<td>Average absolute percentage error in within(out) sample fore(back)casts</td>
</tr>
<tr>
<td>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Roots</td>
<td>(Non)seasonal (AR)MA roots</td>
</tr>
<tr>
<td>MLESummary</td>
<td>Estimation summary</td>
</tr>
<tr>
<td>ForecastCL</td>
<td>Forecasts, standard errors, and confidence limits</td>
</tr>
<tr>
<td>MV1</td>
<td>Original series adjusted for missing value regressors</td>
</tr>
<tr>
<td>A6</td>
<td>RegARIMA trading day component</td>
</tr>
<tr>
<td>A8</td>
<td>RegARIMA combined outlier component</td>
</tr>
<tr>
<td>A8AO</td>
<td>RegARIMA AO outlier component</td>
</tr>
<tr>
<td>A8LS</td>
<td>RegARIMA level change outlier component</td>
</tr>
<tr>
<td>A8TC</td>
<td>RegARIMA temporary change outlier component</td>
</tr>
<tr>
<td>B1</td>
<td>Prior adjusted or original series</td>
</tr>
<tr>
<td>C17</td>
<td>Final weight for irregular components</td>
</tr>
<tr>
<td>C20</td>
<td>Final extreme value adjusted factors</td>
</tr>
<tr>
<td>D1</td>
<td>Modified original data, D iteration</td>
</tr>
<tr>
<td>D7</td>
<td>Preliminary trend cycle, D iteration</td>
</tr>
<tr>
<td>D8</td>
<td>Final unmodified S-I ratios</td>
</tr>
<tr>
<td>D8A</td>
<td>Seasonality tests</td>
</tr>
<tr>
<td>D9</td>
<td>Final replacement values for extreme S-I ratios</td>
</tr>
<tr>
<td>D9A</td>
<td>Moving seasonality ratio</td>
</tr>
<tr>
<td>D10</td>
<td>Final seasonal factors</td>
</tr>
<tr>
<td>D10D</td>
<td>Final seasonal difference</td>
</tr>
<tr>
<td>D11</td>
<td>Final seasonally adjusted series</td>
</tr>
<tr>
<td>D12</td>
<td>Final trend cycle</td>
</tr>
<tr>
<td>D13</td>
<td>Final irregular series</td>
</tr>
<tr>
<td>D16</td>
<td>Combined adjustment factors</td>
</tr>
<tr>
<td>D16B</td>
<td>Final adjustment differences</td>
</tr>
<tr>
<td>D18</td>
<td>Combined calendar adjustment factors</td>
</tr>
<tr>
<td>E4</td>
<td>Ratios of annual totals</td>
</tr>
<tr>
<td>E5</td>
<td>Percent changes in original series</td>
</tr>
<tr>
<td>E6</td>
<td>Percent changes in final seasonally adjusted series</td>
</tr>
<tr>
<td>E7</td>
<td>Differences in final trend cycle</td>
</tr>
<tr>
<td>F2A-I</td>
<td>Summary measures</td>
</tr>
<tr>
<td>F3</td>
<td>Quality assessment statistics</td>
</tr>
<tr>
<td>F4</td>
<td>Day of the week trading day component factors</td>
</tr>
<tr>
<td>G</td>
<td>Spectral analysis</td>
</tr>
</tbody>
</table>
Concepts: Tabular Output and the TEMPLATE Procedure

Viewing the Contents of a Table Definition

To view the contents of a table definition, you can use the SAS windowing environment, the command line, or the TEMPLATE procedure.

1. 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 on SASHELP.TMPLMST to view the contents of that item store or directory.
   4. Double-click on a directory to view the list of subdirectories and table templates that you wish to view. For example, the Base SAS table definition Summary is the default template store for the summary tables created in the MEANS and SUMMARY procedures. Double-click on the Base directory, and then double-click on the Summary table.

2. Using the Command Line
   1. To view the Templates window, submit the following command:

```
   odestemplates
```

   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 definitions that SAS provides, double-click the item store that contains a table definition, such as Base.
   4. Right-click the table definition, such as Summary, and select Open. The table definition is displayed in the Template Browser window.

3. Using the TEMPLATE Procedure
   1. The SOURCE statement writes the source code for the specified definition to the SAS log. For example, if you want to view the source code for all the objects in Base SAS, submit the following code.

```
proc template;
   source base;
run;
```

How Are Values in Table Columns Justified?

The process of justifying the values in columns in a listing output is determined by the format of the variable and the values of two attributes: JUST= and JUSTIFY=. It is a three-step process:
1. ODS puts the value into the format for the column. Character variables are left-justified within their format fields; numeric variables are right-justified.

2. ODS justifies the entire format field within the column width according to the value of the JUST= attribute for the column, or, if that attribute is not set, JUST= for the table. For example, if you right-justify the column, the format field is placed as far to the right as possible. However, the placement of the individual numbers and characters within the field does not change. Thus, decimal points remain aligned. If the column and the format field have the same width, then JUST= has no apparent effect because the format field occupies the entire column.

3. If you specify JUSTIFY=ON for the column or the table, ODS justifies the values within the column without regard to the format field. By default, JUSTIFY=OFF.

For example, consider this set of values:

123.45
234.5
.
987.654

If the values are formatted with a 6.2 format and displayed in a column with a width of 6, they appear this way, regardless of the value of JUST= (asterisks indicate the width of the column):

```
******
123.45
234.50
.
987.65
```

If the width of the column increases to 8, then the value of JUST= does affect the placement of the values because the format field has room to move within the column. Notice that the decimal points remain aligned but that the numbers shift in relation to the column width.

```
just=left  just=center  just=right

******  ********  ********
123.45  123.45    123.45
234.50  234.50    234.50
.       .         .
987.65  987.65    987.65
```

Now, if you add JUSTIFY=ON, then the values are formatted within the column without regard to the format width. The results are as follows:

```
just=on  just=on  just=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
```
If the value of JUST= D, then values are aligned by the decimal point.

<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>

All destinations except LISTING justify the values in columns as if JUSTIFY=ON for JUST=R and JUST=L.

---

**How Are Values in Table Columns Formatted?**

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 do not specify a FORMAT= option, then PROC TEMPLATE uses the format that the data component provides. If the data component does not provide a format, then PROC TEMPLATE uses:
   - best8. for integers
   - 12.3 for doubles
   - the length of the variable for character variables

2. If a format width is specified in the FORMAT= option, then it will take precedence over the FORMAT_WIDTH= and FORMAT_NDEC= options.

3. If you specify a decimal width with the FORMAT= and FORMAT_NDEC= options, then PROC TEMPLATE uses the format that you specified with the FORMAT= option.

4. If you specify a format width with the FORMAT= and FORMAT_WIDTH= options, then PROC TEMPLATE uses the format that you specified with FORMAT= option.

The formatting attributes of a column can be determined by the data component or the column definition. The following table summarizes the behavior of the column formatting attributes based on which attributes the column definition provides.

<table>
<thead>
<tr>
<th>Column definition provides</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 definition; 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 definition.</td>
</tr>
<tr>
<td>format name, width, and number of decimal places</td>
<td>all three are determined by the column definition.</td>
</tr>
</tbody>
</table>
Column definition provides | Result
--- | ---
width | no name is specified; width is determined by the column definition; number of decimal places is determined by the data component.
number of decimal places | no name is specified; width is determined by the data component; number of decimal places is determined by the column definition.

Examples: Modifying Tabular Output by Using the TEMPLATE Procedure

Example 1: Editing a Table Definition that a SAS Procedure Uses

PROC TEMPLATE features:
   EDIT statement
   Header attributes
      JUST=
      STYLE=
   Table attributes
      DOUBLE_SPACE=
      OVERLINE=
      UNDERLINE=

Other ODS features:
   ODS HTML statement
   ODS SELECT statement

Data set: STATEPOP

Program Description

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

This example customizes the table definition for the Moments output object from PROC UNIVARIATE. The first program uses the table definition that SAS supplies to generate both Listing output and HTML output of the Moments object.

The second program
   □ creates and edits a copy of the default table definition.
   □ edits a header within the table definition.
sets column attributes to enhance the appearance of both the HTML and the Listing output.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649. △

Program 1: Using the Default Table Definition that SAS Provides

Set the SAS system options. The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file `defaultmoments-body.htm` in the current directory. Some browsers require an extension of `.htm` or `.html` on the filename.

```
ods html body='defaultmoments-body.htm';
```

Select the output objects for the report. The ODS SELECT statement sends one output object, Moments, to the open ODS destinations. Both the Listing and the HTML destinations are open. (To learn the names of the output objects, run the procedure with the ODS TRACE ON statement in effect. See “Example” on page 200.)

```
ods select moments;
```

Compute the descriptive statistics for one variable. PROC UNIVARIATE computes the univariate statistics for one variable, Citypop_90. It uses the default table definition, `base.univariate.moments` from the template store `sashelp.tmplmst`.

```
proc univariate data=statepop mu0=3.5;
  var citypop_90;
  title 'Default Moments Table';
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. You must close the destination before you can view the output with a browser.

```
ods html close;
```
**Default Listing Output**

**Output 10.3** Listing Output from PROC UNIVARIATE (Default Moments Table)

```
Default Moments Table

The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

Moments

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>51</td>
<td>Sum Weights</td>
</tr>
<tr>
<td>Mean</td>
<td>3.87701961</td>
<td>Sum Observations</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>5.16465302</td>
<td>Variance</td>
</tr>
<tr>
<td>Skewness</td>
<td>2.87109259</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>Uncorrected SS</td>
<td>2100.27737</td>
<td>Corrected SS</td>
</tr>
<tr>
<td>Coeff Variation</td>
<td>133.21194</td>
<td>Std Error Mean</td>
</tr>
</tbody>
</table>
```

**HTML Output from PROC UNIVARIATE (Default Moments Table)**

**Display 10.5** Default HTML Output (Viewed with Microsoft Internet Explorer)

```
Default Moments Table

The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

Moments

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>51</td>
<td>Sum Weights</td>
</tr>
<tr>
<td>Mean</td>
<td>3.87701961</td>
<td>Sum Observations</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>5.16465302</td>
<td>Variance</td>
</tr>
<tr>
<td>Skewness</td>
<td>2.87109259</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>Uncorrected SS</td>
<td>2100.27737</td>
<td>Corrected SS</td>
</tr>
<tr>
<td>Coeff Variation</td>
<td>133.21194</td>
<td>Std Error Mean</td>
</tr>
</tbody>
</table>
```

**Program 2: Using a Customized Table Definition**

**Specify the search path in order to locate the table definition.** The ODS PATH statement specifies which locations to search for definitions 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 definition `base.univariate.moments`. The EDIT statement looks in the available template stores for a table definition 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 definitions that SAS provides. Because the EDIT statement can read this definition, this is the one that it uses. The program does not specify a destination for the edited definition, so PROC TEMPLATE writes to the first template store in the path that it can write to, which is SASUSER.TEMPLAT. Therefore, it creates a table definition of the same name as the original one in SASUSER.TEMPLAT. (See “ODS PATH Statement” on page 149).

(To learn the name of the table definition that a procedure uses, run the procedure with the ODS TRACE ON statement in effect. See “Example” on page 200).

``` 
proc template; 
  edit base.univariate.moments; 
```

Specify changes to the Moments output object. These three table attributes affect the presentation of the `Moments` output object in the Listing output. They have no effect on its presentation in the HTML output. `DOUBLE_SPACE= double spaces between the rows of the output object. OVERLINE= and UNDERLINE= draw a continuous line before the first row of the table and after the last row of the table.`

``` 
  double_space=on; 
  underline=on; 
  overline=on; 
```

Modify a table element. This EDIT statement edits the table element `head` within the table definition.

``` 
  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 definition, `styles.default`. Many procedures, including PROC UNIVARIATE, use this style element to produce headers for tables and columns. (For information on viewing a style definition, see “What Style Definitions Are Shipped with SAS Software?” on page 30.) In this case, the `STYLE=` attribute specifies green for the foreground color and italic for the font style. All other attributes that are included in `header` remain in effect. The `STYLE=` attribute affects only the HTML output.

``` 
  style=header{foreground=green font_style=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 definition. The first END statement ends the editing of the table element **head**. The second END statement ends the editing of the table **base.univariate.moments**.

```
end;
end;
run;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file **custommoments-body.htm** in the current directory. Some browsers require an extension of .htm or .html on the filename.

```
ods html body=`custommoments-body.htm';
```

Select the output objects for the report. The ODS SELECT statement sends one output object, **Moments**, to the open ODS destinations. Both the Listing and the HTML destinations are open. (To learn the names of the output objects, run the procedure with the ODS TRACE ON statement in effect. See “Example” on page 200.)

```
ods select moments;
```

Compute the descriptive statistics for one variable. PROC UNIVARIATE computes the univariate statistics for one variable, CityPop_90. This is the same PROC UNIVARIATE step that was used in “Program 1: Using the Default Table Definition that SAS Provides” on page 516. 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 definition. 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 definition that is supplied by SAS, you would have to change the path with the ODS PATH statement (see “ODS PATH Statement” on page 149).

```
proc univariate data=statepop mu0=3.5;
   var citypop_90;
   title ‘Custom Moments Table’;
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. You must close the destination before you can view the output with a browser.

```
ods html close;
```
Customized Listing Output

Output 10.4  Listing Output (Customized Moments Table) from PROC UNIVARIATE

<table>
<thead>
<tr>
<th>Custom Moments Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>The UNIVARIATE Procedure</td>
</tr>
<tr>
<td>Variable: CityPop_90 (1990 metropolitan pop in millions)</td>
</tr>
<tr>
<td>Moments</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>51</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>3.87701961</td>
</tr>
<tr>
<td>Std Deviation</td>
</tr>
<tr>
<td>5.16465302</td>
</tr>
<tr>
<td>Skewness</td>
</tr>
<tr>
<td>2.87109259</td>
</tr>
<tr>
<td>Uncorrected SS</td>
</tr>
<tr>
<td>2100.27737</td>
</tr>
<tr>
<td>Coeff Variation</td>
</tr>
<tr>
<td>133.21194</td>
</tr>
</tbody>
</table>

Customized HTML Output

Display 10.6  Customized HTML Output (Customized Moments Table) from PROC UNIVARIATE (Viewed with Microsoft Internet Explorer)

Example 2: Comparing the EDIT Statement with the DEFINE TABLE Statement

PROC TEMPLATE features:
EDIT statement
COLUMN statement
DEFINE statement
   STYLE= attribute
NOTES statement
DYNAMIC statement

Other ODS features:
   ODS PATH statement
   ODS HTML statement
   ODS HTML CLOSE statement

Program Description

This example compares the use of an EDIT statement with a DEFINE TABLE statement for the same table definition. The first program uses the EDIT statement to change the Base.Summaries table definition. The foreground color of the NOBS column is changed to green. The other definitions and attributes of the Base.Summaries table definition remain the same. The second program uses the DEFINE TABLE statement to define a new table using the same name, Base.Summaries. The NOBS column is the only column defined in the new table definition. When the PROC SUMMARY step executes, only the NOBS column is printed. The only style attribute that is used to format the column is the foreground=green attribute.

Program 1

Edit the existing table definition Base.Summaries. The ODS PATH statement specifies which item store to search first for the table definition. The EDIT statement edits the table definition Base.Summaries. The modified table definition Base.Summaries is written to the WORK.TEMPLAT item store.

   Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

ods path work.templat (update) sashelp.tmplmst (read);
proc template;
   edit Base.Summary;
      edit nobs;
      style={foreground=green};
   end;
end;
run;
ods html file="temp.html";
proc summary data=sashelp.class print;
class age;
run;
ods html close;
The column labeled *AGE* remains in the output because *AGE* is defined as a dynamic variable which is passed to the original **Base.Summary** table definition and *AGE* is specified as the CLASS variable. The attributes of the NOBS column are modified in the EDIT statement where the NOBS column is defined.
Output 10.5  Base.Summary Table Definition Modified by the EDIT Statement

The modified Base.Summary table definition changes the foreground color of the NOBS column to green. The vertical alignment and heading of the NOBS column, and the other table attributes, are retained from the default table definition and stay the same. To view the Base.Summary table definition created by Program 1, follow these steps.

1  Submit the following command in the command bar:

    odestemplates

2  Double-click the item store WORK.TEMPLAT.

3  Double-click the item store Base.

4  Right-click the table definition Summary and select Open. The table definition Base.Summary is displayed in the Template Browser window.

proc template;
    define table Base.Summary / store = SASUSER.TEMPLAT;
        notes "Summary table for MEANS and SUMMARY";
        dynamic one_var one_var_label one_var_name clmpct;
        column class nobs id type ways (varname) (label) (min) (max) (range) (n)
            (nmiss) (sumwgt) (sum) (mean) (uss) (css) (var) (stddev) (cv) (stderr)
            (t) (probt) (lclm) (uclm) (skew) (kurt) (median) (mode) (q1) (q3) (qrange)
            (p1) (p5) (p10) (p25) (p50) (p75) (p90) (p95) (p99);
        header =;
        define p99;
            header = "99th Pctl";
            generic;
        end;
        define p95;
            header = "95th Pctl";
            generic;
        end;
        define p90;
            header = "90th Pctl";
            generic;
        end;
        define p75;
            header = "75th Pctl";
            generic;
        end;
        define p50;
            header = "50th Pctl";
            generic;
        end;
        define p25;
            header = "25th Pctl";
            generic;
        end;
        define p10;
            header = "10th Pctl";
            generic;
        end;
        define p5;
            header = "5th Pctl";
            generic;
        end;
        define p1;
            header = "1st Pctl";
            generic;
        end;
        define qrange;
            header = "Quartile Range";
            generic;
        end;
        define q3;
            header = "Upper Quartile";
            generic;
        end;
        define q1;
            header = "Lower Quartile";
            generic;
        end;
    end;
end;
define mode;
  header = "Mode";
  generic;
end;
define median;
  header = "Median";
  generic;
end;
define kurt;
  header = "Kurtosis";
  generic;
end;
define skew;
  header = "Skewness";
  generic;
end;
define uclm;
  define header huclm;
    text "Upper " clmpt BEST8. %nrstr("/\% CL for Mean");
    split = "/";
  end;
  header = huclm;
  generic;
end;
define lclm;
  define header hlclm;
    text "Lower " clmpt BEST8. %nrstr("/\% CL for Mean");
    split = "/";
  end;
  header = hlclm;
  generic;
end;
define probt;
  parent = Common.ParameterEstimates.Probt;
  generic;
end;
define t;
  parent = Common.ParameterEstimates.tValue;
  generic;
end;
define stderr;
  header = "Std Error";
  parent = Common.ParameterEstimates.StdErr;
  generic;
end;
define cv;
  header = "Coeff of Variation";
  generic;
end;
define stddev;
  header = "Std Dev";
  generic;
end;
define var;
  header = "Variance";
  generic;
end;
define css;
  define header hcss;
    text2 "CSS";
    text "Corrected SS";
  end;
  header = hcss;
  generic;
end;
define uss;
  define header huss;
    text2 "USS";
    text "Uncorrected SS";
  end;
  header = huss;
  generic;
end;
define mean;
    header = "Mean";
    generic;
end;
define sum;
    header = "Sum";
    generic;
end;
define sumwgt;
    header = "Sum Wgts";
    generic;
end;
define nmiss;
    header = "N Miss";
    generic;
end;
define n;
    header = "N";
    generic;
end;
define range;
    header = "Range";
    generic;
end;
define max;
    define header hmax;
        text2 "Max";
        text "Maximum";
        end;
    header = hmax;
    generic;
end;
define min;
    define header hmin;
        text2 "Min";
        text "Minimum";
        end;
    header = hmin;
    generic;
end;
define label;
    header = "Label";
    id;
    generic;
end;
define varname;
    header = "Variable";
    id;
    generic;
end;
define ways;
    header = "Ways";
    vjust = T;
    id;
end;
define type;
    header = "Type";
    vjust = T;
    id;
end;
define id;
    vjust = T;
    id;
    generic;
end;
Program 2

Define the table Base.Summary. The ODS PATH statement specifies which item store to search first for the table definition. The DEFINE TABLE statement creates a new table definition Base.Summary. The new table definition Base.Summary is written to the WORK.TEMPLAT item store.

```sas
ods path work.templat (update) sashelp.tmplmst (read);
proc template;
  define table Base.Summary;
    notes "Summary table for MEANS and SUMMARY";
    dynamic clmpt one_var_name one_var_label one_var;
    column class nosb 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 nosb;
      style={foreground=green};
    end;
  end;
run;
ods html file="temp.html";
```
proc summary data=sashelp.class print;
class age;
run;

ods html close;

Display 10.8  HTML Output that Uses the Table Definition Base.Summary.

The column labeled AGE is missing because it was not defined in the new table definition Base.Summary. The new table definition only defined the NOBS column with a green foreground and no column headings.
Output 10.6  Base.Summary Table Definition  Created by the DEFINE TABLE Statement

The **Base.Summary** table definition defines the foreground color of the NOBS column to green. Because the vertical alignment and heading of the NOBS column, and the other table attributes, are not defined, they are no longer part of the **Base.Summary** table definition. To view the table definition **Base.Summary** created by Program 2, follow these steps.

1. Submit the following command:

   `odstemplates`

2. Double-click the item store **WORK.TEMPLAT**.

3. Double-click the item store **Base**.

4. Right-click the table definition **Summary** and select **Open**. The table definition **Base.Summary** is displayed in the Template Browser window.

```proc template;
define table Base.Summary / store = WORK.TEMPLAT;
  notes "Summary table for MEANS and SUMMARY";
  dynamic clmpct one_var_name one_var_label one_var;
  column class nobs id type ways (varname) (label) (min) (max) (range) (n) (nmiss) (sumwgt) (sum) (mean) (uss) (css) (var) (stddev) (cv) (stderr) (t) (probt) (lclm) (uclm) (skew) (kurt) (median) (mode) (q1) (q3) (qrange) (p1) (p5) (p10) (p25) (p50) (p75) (p90) (p95) (p99);
  define nobs;
    style = {
      foreground = green
    };
  end;
end;
run;```

**Example 3: Creating a New Table Definition**

**PROC TEMPLATE** features:

Table attributes:
- **DOUBLE_SPACE=**
- **OVERLINE=**
- **UNDERLINE=**

**DEFINE TABLE statement**:
- **COLUMN** statement
- **DEFINE** statement (for columns)
  - **GENERIC=** attribute
  - **HEADER=** attribute
  - **ID=** attribute
  - **STYLE=** attribute
  - **VJUST=** attribute

**DEFINE** statement (for headers)
- **TEXT** statement
- **STYLE=** attribute
- **SPACE=** attribute

**DEFINE FOOTER** statement
HEADER statement
MVAR statement

Other ODS features:
    ODS HTML statement
    FILE statement with ODS= option
    PUT statement with _ODS_ argument

Program Description

This example creates a custom table definition for an output data set that PROC MEANS produces.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.

Program 1: Producing an Output Data Set with PROC MEANS

Set the SAS system options. The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output.

```sas
options nodate pageno=1 pagesize=60 linesize=72;
```

Create formats for the variables Year and School. PROC FORMAT creates formats for Year and School.

```sas
proc format;
    value yrFmt . = “All”;
    value $schFmt ‘ ’ = “All ”;
run;
```

The data set Charity contains information about students' volunteer work for charity. The variables give the name of the school, the year of the fundraiser, the first name of each student, the amount of money that each student raised, and the number of hours that each student volunteered. The RETAIN statement and two sum statements compute the minimum and maximum values of Year. The CALL SYMPUT statements store these values in the macro variables FIRST_YEAR and LAST_YEAR. A DATA step creates this data set.

```sas
data Charity;
    input School $ 1-7 Year 9-12 Name $ 14-20 moneyRaised 22-26
        hoursVolunteered 28-29;
    format moneyRaised dollar8.2;
    format hoursVolunteered f3.0;
    format Year yrFmt.;
    format School schFmt.;
    label School = “Schools”;
    label Year = “Years”; retain yearmin yearmax;
    yearmin=min(yearmin,year);
    yearmax=max(yearmax,year);
```

call symput('first_year', put(yearmin,4.));
call symput('last_year', put(yearmax,4.));
datalines;
Monroe 1992 Allison 31.65 19
Monroe 1992 Barry  23.76 16
Monroe 1992 Candace 21.11  5
Monroe 1992 Danny   6.89  23
Monroe 1992 Edward  53.76  31
Monroe 1992 Fiona   48.55  13
Monroe 1992 Gert    24.00  16
Monroe 1992 Harold  27.55  17
Monroe 1992 Ima     15.98  9
Monroe 1992 Jack    20.00  23
Monroe 1992 Katie   22.11  2
Monroe 1992 Lisa    18.34  17
Monroe 1992 Tonya   55.16  40
Monroe 1992 Max     26.77  34
Monroe 1992 Ned     28.43  22
Monroe 1992 Opal    32.66  14
Monroe 1993 Patsy   18.33  18
Monroe 1993 Quentin 16.89  15
Monroe 1993 Randall 12.98  17
Monroe 1993 Sam     15.88  5
Monroe 1993 Tyra    21.88  23
Monroe 1993 Myrtle  47.33  26
Monroe 1993 Frank   41.11  22
Monroe 1993 Cameron 65.44  14
Monroe 1993 Vern    17.89  11
Monroe 1993 Wendell 23.00  10
Monroe 1993 Bob     26.88  6
Monroe 1993 Leah    28.99  23
Monroe 1994 Becky   30.33  26
Monroe 1994 Sally   35.75  27
Monroe 1994 Edgar   27.11  12
Monroe 1994 Dawson  17.24  16
Monroe 1994 Lou     5.12   16
Monroe 1994 Damien  18.74  17
Monroe 1994 Mona    27.43  7
Monroe 1994 Della   56.78  15
Monroe 1994 Monique 29.88  19
Monroe 1994 Carl    31.12  25
Monroe 1994 Reba    35.16  22
Monroe 1994 Dax     27.65  23
Monroe 1994 Gary    23.11  15
Monroe 1994 Suzie   26.65  11
Monroe 1994 Benito  47.44  18
Monroe 1994 Thomas  21.99  23
Monroe 1994 Annie   24.99  27
Monroe 1994 Paul    27.98  22
Monroe 1994 Alex    24.00  16
Monroe 1994 Lauren  15.00  17
Monroe 1994 Julia   12.98  15
Monroe 1994 Keith   11.89  19
Monroe 1994 Jackie  26.88  22
<table>
<thead>
<tr>
<th>School</th>
<th>Year</th>
<th>Last Name</th>
<th>First Name</th>
<th>Score</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monroe</td>
<td>1994</td>
<td>Pablo</td>
<td></td>
<td>13.98</td>
<td>28</td>
</tr>
<tr>
<td>Monroe</td>
<td>1994</td>
<td>L.T.</td>
<td></td>
<td>56.87</td>
<td>33</td>
</tr>
<tr>
<td>Monroe</td>
<td>1994</td>
<td>Willard</td>
<td></td>
<td>78.65</td>
<td>24</td>
</tr>
<tr>
<td>Monroe</td>
<td>1994</td>
<td>Kathy</td>
<td></td>
<td>32.88</td>
<td>11</td>
</tr>
<tr>
<td>Monroe</td>
<td>1994</td>
<td>Abby</td>
<td></td>
<td>35.88</td>
<td>10</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Arturo</td>
<td></td>
<td>34.98</td>
<td>14</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Grace</td>
<td></td>
<td>27.55</td>
<td>25</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Winston</td>
<td></td>
<td>23.88</td>
<td>22</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Vince</td>
<td></td>
<td>12.88</td>
<td>21</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Claude</td>
<td></td>
<td>15.62</td>
<td>5</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Mary</td>
<td></td>
<td>28.99</td>
<td>34</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Abner</td>
<td></td>
<td>25.89</td>
<td>22</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Jay</td>
<td></td>
<td>35.89</td>
<td>35</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Alicia</td>
<td></td>
<td>28.77</td>
<td>26</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Freddy</td>
<td></td>
<td>29.00</td>
<td>27</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Eloise</td>
<td></td>
<td>31.67</td>
<td>25</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Jenny</td>
<td></td>
<td>43.89</td>
<td>22</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Thelma</td>
<td></td>
<td>52.63</td>
<td>21</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Tina</td>
<td></td>
<td>19.67</td>
<td>21</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1992</td>
<td>Eric</td>
<td></td>
<td>24.89</td>
<td>12</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Bubba</td>
<td></td>
<td>37.88</td>
<td>12</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>G.L.</td>
<td></td>
<td>25.89</td>
<td>21</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Bert</td>
<td></td>
<td>28.89</td>
<td>21</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Clay</td>
<td></td>
<td>26.44</td>
<td>21</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Leaann</td>
<td></td>
<td>27.17</td>
<td>17</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Georgia</td>
<td></td>
<td>38.90</td>
<td>11</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Bill</td>
<td></td>
<td>42.23</td>
<td>25</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Holly</td>
<td></td>
<td>18.67</td>
<td>27</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Benny</td>
<td></td>
<td>19.09</td>
<td>25</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Cammie</td>
<td></td>
<td>28.77</td>
<td>28</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Amy</td>
<td></td>
<td>27.08</td>
<td>31</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Doris</td>
<td></td>
<td>22.22</td>
<td>24</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Robbie</td>
<td></td>
<td>19.80</td>
<td>24</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Ted</td>
<td></td>
<td>27.07</td>
<td>25</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Sarah</td>
<td></td>
<td>24.44</td>
<td>12</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Megan</td>
<td></td>
<td>28.89</td>
<td>11</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Jeff</td>
<td></td>
<td>31.11</td>
<td>12</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Taz</td>
<td></td>
<td>30.55</td>
<td>11</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>George</td>
<td></td>
<td>27.56</td>
<td>11</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1993</td>
<td>Heather</td>
<td></td>
<td>38.67</td>
<td>15</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Nancy</td>
<td></td>
<td>29.90</td>
<td>26</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Rusty</td>
<td></td>
<td>30.55</td>
<td>28</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Mimi</td>
<td></td>
<td>37.67</td>
<td>22</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>J.C.</td>
<td></td>
<td>23.33</td>
<td>27</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Clark</td>
<td></td>
<td>27.90</td>
<td>25</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Rudy</td>
<td></td>
<td>27.78</td>
<td>23</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Samuel</td>
<td></td>
<td>34.44</td>
<td>18</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Forrest</td>
<td></td>
<td>28.89</td>
<td>26</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Luther</td>
<td></td>
<td>72.22</td>
<td>24</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Trey</td>
<td></td>
<td>6.78</td>
<td>18</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Albert</td>
<td></td>
<td>23.33</td>
<td>19</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Che-Min</td>
<td></td>
<td>26.66</td>
<td>33</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Preston</td>
<td></td>
<td>32.22</td>
<td>23</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1994</td>
<td>Larry</td>
<td></td>
<td>40.00</td>
<td>26</td>
</tr>
</tbody>
</table>
Compute the descriptive statistics, and specify the options and subgroups for analysis. This PROC MEANS step analyzes the data for the one-way combination of the class variables and across all observations. It creates an output data set that includes variables for the total and average amount of money raised. The data set also includes new variables for the top three amounts of money raised, the names of the three students who raised the money, the years when the students raised the money, and the schools that the students attended.

```
proc means data=Charity desc=Types char=Type noprint;
  class School Year;
  var moneyRaised;
  types () School year;
  output out=top3list sum= mean=
    idgroup ( max(moneyRaised) out[3](moneyRaised name school year)= )
    / autoname;
run;
```

Print the report. This PROC PRINT step generates traditional Listing output of the output data set that PROC MEANS created.

```
proc print data=top3list nobs;
  title 'Simple PROC PRINT of the Output Data Set';
run;
```

### Listing Output from PROC PRINT

#### Output 10.7 PROC PRINT Listing Output from PROC MEANS

<table>
<thead>
<tr>
<th>School</th>
<th>Year</th>
<th><em>TYPE</em> <em>FREQ</em></th>
<th>money Raised Sum</th>
<th>money Raised Mean</th>
<th>money Raised1</th>
<th>money Raised2</th>
<th>money Raised3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kennedy</td>
<td>All</td>
<td>10</td>
<td>53</td>
<td>$1575.95</td>
<td>$29.73</td>
<td>$72.22</td>
<td>$52.63</td>
</tr>
<tr>
<td>Monroe</td>
<td>All</td>
<td>10</td>
<td>56</td>
<td>$1616.80</td>
<td>$28.87</td>
<td>$78.65</td>
<td>$65.44</td>
</tr>
<tr>
<td>All</td>
<td>1992</td>
<td>01</td>
<td>31</td>
<td>$892.92</td>
<td>$28.80</td>
<td>$55.16</td>
<td>$53.76</td>
</tr>
<tr>
<td>All</td>
<td>1993</td>
<td>01</td>
<td>32</td>
<td>$907.92</td>
<td>$28.37</td>
<td>$65.44</td>
<td>$47.33</td>
</tr>
<tr>
<td>All</td>
<td>1994</td>
<td>01</td>
<td>46</td>
<td>$1391.91</td>
<td>$30.26</td>
<td>$78.65</td>
<td>$72.22</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>00</td>
<td>109</td>
<td>$3192.75</td>
<td>$29.29</td>
<td>$78.65</td>
<td>$72.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name_1</th>
<th>Name_2</th>
<th>Name_3</th>
<th>School_1</th>
<th>School_2</th>
<th>School_3</th>
<th>Year_1</th>
<th>Year_2</th>
<th>Year_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameron</td>
<td>Myrtle</td>
<td>Bill</td>
<td>Monroe</td>
<td>Monroe</td>
<td>Kennedy</td>
<td>1993</td>
<td>1993</td>
<td>1993</td>
</tr>
<tr>
<td>Willard</td>
<td>Luther</td>
<td>Cameron</td>
<td>Monroe</td>
<td>Kennedy</td>
<td>Monroe</td>
<td>1994</td>
<td>1994</td>
<td>1993</td>
</tr>
</tbody>
</table>
Set the SAS system options. The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output.

```
options nodate pageno=1 pagesize=60 linesize=72;
```

Create the HTML output and specify the name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file `topn-body.htm` in the current directory. Some browsers require an extension of `.htm` or `.html` on the filename.

```
ods html body='topn-body.htm';
```

Create the table definition `means.topn` The DEFINE statement creates the table definition `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 definition 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 definition one below the other in the output object. Variables are assigned a column definition 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 definition. These three table attributes affect the presentation of the output object in the Listing output. They have no effect on its presentation in the HTML output. DOUBLE_SPACE= double spaces the rows of the output object. OVERLINE= and UNDERLINE= draw a continuous line before the first row of the table and after the last row of the table.

```
double_space=on;
  overline=on;
  underline=on;
```

Specify the two table headers and the order in which they appear in the report. The HEADER statement declares `table_header_1` and `table_header_2` as headers in the table and specifies the order in which the headers appear in the output object.

```
header table_header_1 table_header_2;
```
Create the table element `table_header_1`. The DEFINE statement and its substatement and attribute define `table_header_1`. The TEXT statement specifies the text of the header. The STYLE= attribute alters the style element that displays the table header. The style element `header` is defined in the default style definition, `styles.default`. (For information on viewing a style definition, see “What Style Definitions Are Shipped with SAS Software?” on page 30.) In this case, the STYLE= attribute specifies a large font size. All other attributes that are included in `header` remain in effect. This attribute affects only the HTML output.

The END statement ends the header definition.

```sas
define table_header_1;
  text "Top Three Fund Raisers";
  style=header{font_size=6};
end;
```

Create the table element `table_header_2`. The DEFINE statement and its substatement and attribute define `table_header_2`. The TEXT statement uses text and the macro variables `FIRST_YEAR` and `LAST_YEAR` to specify the contents of the header. When ODS binds the data component to the table definition (in the DATA step that follows), it will resolve the values of the macro variables `FIRST_YEAR` and `LAST_YEAR`. The table definition 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 definition.

```sas
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 definition (in the DATA step that follows), it will resolve the value of the macro variable `SYSDATE9`. The table definition itself contains a reference to the macro variable. The SPLIT= attribute specifies the asterisk as the split character. This prevents the header from splitting at the open parenthesis. If no split character is specified, then ODS interprets the nonalphabetic, leading character as the split character (see the discussion of `text-specification(s)` in “TEXT Statement” on page 409.) Alternatively, you can 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 definition, `styles.default`. (For information on viewing a style definition, see “Viewing the Contents of a Style Definition” on page 319.) 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 definition.

```sas
define footer table_footer;
  text "(report generated on " sysdate9 ")";
  split="*";
  style=header{font_size=2};
end;
```
**Create the column class.** The DEFINE statement and its attributes create the column definition `class` (The COLUMN statement earlier in the program declared `class` as a column.) GENERIC= specifies that multiple variables can use the same column definition. 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 definition, which is the style definition that is being used. STYLE= affects only the HTML output.

The END statement ends the definition.

Notice that, unlike subsequent column definitions, this column definition does not include a header. This is because the same header is not appropriate for all the variables that use this column definition. 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.

```plaintext
define class;
  generic=on;
  id=on;
  vjust=top;
  style=data;
end;
```

**Create six additional columns.** Each of these DEFINE statements and its attributes creates a column definition. GENERIC= specifies that multiple variables can use a column definition (although in the case of `sum` and `mean`, only one variable uses the definition). 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 definition.

```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;
```
Program 2: Building a Custom Table Definition for the TopN Report

Chapter 10

header="School";
end;

define year;
    generic=on;
    header="Year";
end;

End the table definition. This END statement ends the table definition. The RUN statement ends the PROC TEMPLATE step.

end;
run;

Create the data component. This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object. The SET statement reads the data set TOP3LIST that was created with PROC MEANS.

data _null_
set top3list;

Route the DATA step results to ODS and use the means.topn table definition. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information on using the DATA step with ODS, see Chapter 3, “Output Delivery System and the DATA Step,” on page 39.) The TEMPLATE= suboption tells ODS to use the table definition named means.topn, which was previously created with PROC TEMPLATE.

file print ods = (  
    template='means.topn'

Specify the column definition to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table definition. For example, the first columnspecification specifies that the first column of the output object contains the values of the variable SCHOOL and that it uses the column definition named class. GENERIC= must be set to ON in both the table definition and each column assignment in order for multiple variables to use the same column definition.

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.

```latex
put _ods_;
run;
```

Stop the creation of HTML output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser.

```latex
ods html close;
```

**Listing Output for the TopN Report**

Compare this customized output to the PROC PRINT listing output in Output 10.7.

**Output 10.8** Using a Customized Table to Produce Listing Output

<table>
<thead>
<tr>
<th>Schools</th>
<th>Years</th>
<th>Total Dollars Raised</th>
<th>Average Dollars per Student</th>
<th>Average Dollars per Individual Dollars</th>
<th>Student</th>
<th>School</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kennedy</td>
<td>All</td>
<td>$1575.95</td>
<td>$29.73</td>
<td>$72.22</td>
<td>Luther</td>
<td>Kennedy</td>
<td>1994</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$52.63</td>
<td>Thelma</td>
<td>Kennedy</td>
<td>1992</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$43.89</td>
<td>Jenny</td>
<td>Kennedy</td>
<td>1992</td>
</tr>
<tr>
<td>Monroe</td>
<td>All</td>
<td>$1606.80</td>
<td>$28.69</td>
<td>$78.65</td>
<td>Willard</td>
<td>Monroe</td>
<td>1994</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$65.44</td>
<td>Cameron</td>
<td>Monroe</td>
<td>1993</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$56.87</td>
<td>L.T.</td>
<td>Monroe</td>
<td>1994</td>
</tr>
<tr>
<td>All</td>
<td>1992</td>
<td>$882.92</td>
<td>$28.48</td>
<td>$55.16</td>
<td>Tonya</td>
<td>Monroe</td>
<td>1992</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$53.76</td>
<td>Edward</td>
<td>Monroe</td>
<td>1992</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$52.63</td>
<td>Thelma</td>
<td>Kennedy</td>
<td>1992</td>
</tr>
<tr>
<td>All</td>
<td>1993</td>
<td>$907.92</td>
<td>$28.37</td>
<td>$65.44</td>
<td>Cameron</td>
<td>Monroe</td>
<td>1993</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$47.33</td>
<td>Myrtle</td>
<td>Monroe</td>
<td>1993</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$42.23</td>
<td>Bill</td>
<td>Kennedy</td>
<td>1993</td>
</tr>
<tr>
<td>All</td>
<td>1994</td>
<td>$1391.91</td>
<td>$30.26</td>
<td>$78.65</td>
<td>Willard</td>
<td>Monroe</td>
<td>1994</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$72.22</td>
<td>Luther</td>
<td>Kennedy</td>
<td>1994</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$56.87</td>
<td>L.T.</td>
<td>Monroe</td>
<td>1994</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>$3182.75</td>
<td>$29.20</td>
<td>$78.65</td>
<td>Willard</td>
<td>Monroe</td>
<td>1994</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$72.22</td>
<td>Luther</td>
<td>Kennedy</td>
<td>1994</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$65.44</td>
<td>Cameron</td>
<td>Monroe</td>
<td>1993</td>
</tr>
</tbody>
</table>

(report generated on 30JUN2003)
Example 4: Changing a Column without Redefining the Table Definition

PROC TEMPLATE features:
- DEFINE TABLE statement
  - Table attributes:
Other ODS features:
- ODS HTML statement

Program Description

Program

```plaintext
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={foreground=green};

   id;
   end;

   end;
run;

ods html file="tmp.html";

proc summary data=sashelp.class print;
class age;
run;

ods html close;

Example 5: Setting the Style Element for Cells Based on Their Values

PROC TEMPLATE features:
   DEFINE TABLE statement
      NMVAR statement
      NOTES statement
      TRANSLATE-INTO statement
   DEFINE COLUMN statement
      BLANK_DUPS= attribute
      CELLSSTYLE-AS statement
      GENERIC= attribute

Other ODS features:
   ODS HTML statement
      FILE statement with ODS= option
      PUT statement with _ODS_ argument

Data set:   GRAIN_PRODUCTION “Program” on page 97
Format:    $CNTRY.“Program” on page 97

Program Description

This example creates a template that uses different colors and font attributes for the text inside cells, depending on their values.

Note: This example uses filenames that might not be valid in all operating environments. To successfully run the example in your operating environment, you might need to change the file specifications. See Appendix 3, “ODS HTML Statements for Running Examples in Different Operating Environments,” on page 649.
Program

**Set the SAS system options.** The OPTIONS statement controls several aspects of the Listing output. None of these options affects the HTML output. The TITLE statement specifies a title.

```sas
options nodate pageno=1 pagesize=60 linesize=72;
title 'Leading Grain Producers';
```

**Create the table definition shared.cellstyle.** The DEFINE statement creates the table definition `shared.cellstyle` in the first template store in the path that is available to write to. By default, this template store is SASUSER.TEMPLAT.

```sas
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.

```sas
translate _val_=. into 'No data';
```

**Store the information about the table in the table definition.** The NOTES statement provides information about the table. NOTES statements remain a part of the compiled table definition whereas SAS comments do not.

```sas
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 definition 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 definition whereas SAS comments do not. LOW, MEDIUM, and HIGH will contain the values to use as the determinants of the style element that is used to display the cell. The values are provided just before the DATA step that produces the report.

```sas
nmvar low 'Use default style.'
  medium 'Use yellow foreground and bold font weight'
  high 'Use red foreground 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, you should set this attribute as well.

```sas
classlevels=on;
```

**Create the column char_var.** The DEFINE statement and its attributes create the column definition `char_var`. GENERIC= specifies that multiple variables can use the same column definition. 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 definition.

```sas
create the table definition shared.cellstyle.```
define column char_var;
  generic=on;
  blank_dups=on;
end;

Create the column num_var. The DEFINE statement and its attributes create the column definition num_var. GENERIC= specifies that multiple variables can use the same column definition.

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 definition, the value No data is formatted as a character variable (left-justified) within a format field that has the same width as the column.

justify=on;

Specify which style element and style attributes to use for different values in the column. The CELLSTYLE-AS statement specifies the style element and style attributes to use for different values in this column. If a value is less than or equal to the value of the variable LOW, the cell uses the unaltered Data style element. If a value is greater than LOW but less than or equal to the value of MEDIUM, the cell uses the style element Data with a foreground color of green and an italic font. Similarly, other values use a foreground color of yellow or red and combinations of a bold font weight and an italic font style. The CELLSTYLE-AS statement affects only the HTML destination.

The END statement ends the column definition.

    cellstyle _val_ <= low as data,
    _val_ <= medium as data
        {foreground=green font_style=italic},
    _val_ <= high as data
        {foreground=yellow font_weight=bold},
    1 as data
        {foreground=red font_style=italic
            font_weight=bold};
end;

End the table definition. This END statement ends the table definition. The RUN statement ends the PROC TEMPLATE step.

end;
run;

Create the HTML output and specify name of the HTML file. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the external file cellstyle-body.htm in the current directory. Some browsers require an extension of .htm or .html on the filename.

ods html body='cellstyle-body.htm';
Assign values to three macro variables. The %LET statements assign values to the macro variables LOW, MEDIUM, and HIGH.

```latex
%let low=10000;
%let medium=50000;
%let high=100000;
```

Create the data component. This DATA step does not create a data set. Instead, it creates a data component, and, eventually, an output object. The SET statement reads the data set GRAIN_PRODUCTION.

```latex
data _null_;  
set grain_production;
```

Route the DATA step results to ODS and use the \texttt{shared.cellstyle} table definition. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information on using the DATA step with ODS, see Chapter 3, “Output Delivery System and the DATA Step,” on page 39.) The TEMPLATE= suboption tells ODS to use the table definition named \texttt{shared.cellstyle}, which was previously created with PROC TEMPLATE.

```latex
file print ods=(
  template='shared.cellstyle'
);
```

Specify the column definition to use for each variable. The COLUMNS= suboption places DATA step variables into columns that are defined in the table definition. For example, the first column-specification specifies that the first column of the output object contains the values of the variable \texttt{YEAR} and that it uses the column definition named \texttt{char_var}. GENERIC= must be set to ON, both in the table definition and in each column assignment, in order for multiple variables to use the same column definition.

```latex
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.

```latex
put _ods_;  
run;
```

Stop the creation of HTML output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. You must close the destination before you can view the output with a browser.

```latex
ods html close;
```
### Listing Output of a Customized Table

#### Output 10.9  Listing Output

Only the table customizations appear in the Listing output. Table customizations include the suppression of values that do not change from one row to the next and the translation of missing values to **No data**. The style customizations that are specified in the CELLSTYLE-AS statement do not appear in the Listing output.

<table>
<thead>
<tr>
<th>Year</th>
<th>Country</th>
<th>Type</th>
<th>Kilotons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>Brazil</td>
<td>Corn</td>
<td>36,276</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>11,236</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>1,516</td>
</tr>
<tr>
<td></td>
<td>China</td>
<td>Corn</td>
<td>112,331</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>185,226</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>102,207</td>
</tr>
<tr>
<td></td>
<td>India</td>
<td>Corn</td>
<td>9,800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>122,372</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>63,007</td>
</tr>
<tr>
<td></td>
<td>Indonesia</td>
<td>Corn</td>
<td>8,223</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>49,860</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>No data</td>
</tr>
<tr>
<td></td>
<td>United States</td>
<td>Corn</td>
<td>187,300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>7,888</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>59,494</td>
</tr>
<tr>
<td>1996</td>
<td>Brazil</td>
<td>Corn</td>
<td>31,975</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>10,035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>3,302</td>
</tr>
<tr>
<td></td>
<td>China</td>
<td>Corn</td>
<td>119,350</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>190,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>109,000</td>
</tr>
<tr>
<td></td>
<td>India</td>
<td>Corn</td>
<td>8,660</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>120,012</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>62,620</td>
</tr>
<tr>
<td></td>
<td>Indonesia</td>
<td>Corn</td>
<td>8,925</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>51,165</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>No data</td>
</tr>
<tr>
<td></td>
<td>United States</td>
<td>Corn</td>
<td>236,064</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rice</td>
<td>7,771</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheat</td>
<td>62,099</td>
</tr>
</tbody>
</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 6: Setting the Style Element for a Specific Column, Row, and Cell**

**PROC TEMPLATE features:**
- DEFINE STYLE statement
- REPLACE statement
- DEFINE TABLE statement
CELLSYTL–AS statement
DEFINE COLUMN statement
  DEFINE HEADER statement
    TEXT statement
  DEFINE HEADER statement
    TEXT statement

Other ODS features:
  FILE statement with ODS= option
  ODS HTML statement
    STYLE= option
  ODS PDF statement
    STYLE= option
  PUT statement with _ODS_ argument
  ODS TRACE statement

---

Program Description

This example combines a customized style definition with a customized table definition to produce output with a checkerboard pattern of table cells.

Program

Create a new style definition Greenbar. The PROC TEMPLATE statement starts the TEMPLATE procedure. The DEFINE STYLE statement creates a new style definition greenbar.

```
proc template;
  define style greenbar;
```

Specify the parent style definition from which the greenbar style definition inherits its attributes. The PARENT= attribute specifies the style definition 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 definition from the parent style definition, but the background is light green and the foreground is black.

```
replace headersandfooters from cell /
  background=light green
  foreground=black
;
```
End the style definition. The END statement ends the style definition. The RUN statement executes the PROC TEMPLATE step.

```plaintext
end;
run;
```

Create the HTML and PDF output and specify the style definition that you want to use for the output. The ODS HTML statement opens the HTML destination and creates HTML output. It sends all output objects to the file `greenbar.html` in the current directory. The STYLE= option tells ODS to use `greenbar` as the style definition 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 definition when it formats the output.

```plaintext
ods html body="greenbar.html" style=greenbar;
ods pdf file="greenbar.pdf" style=greenbar;
```

Create the table definition Checkerboard. The DEFINE statement creates the table definition Checkerboard in the first template store in the path that is available to write to. By default, this template store is SASUSER.TEMPLAT.

```plaintext
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 magenta in color. Similarly, if both the row and column are even numbered, then the cell is magenta in color. The CELLSTYLE-AS statement has no effect on the LISTING destination because it is changing style elements and style attributes which have no effect in listing output.

```plaintext
  cellstyle mod(_row_,2) && mod(_col_,2) as data{background=magenta},
  not(mod(_row_,2)) && not(mod(_col_,2)) as data{background=magenta},
  1 as data;
```

Create the header definition top. The DEFINE HEADER statement defines the table header top.
The TEXT statement specifies the text of the header Checkerboard Table Definition. The END statement ends the header definition.

```plaintext
define header top;
  text ‘Checkerboard Table Definition’;
end;
```
Create the column definition name. The DEFINE COLUMN statement creates the column definition name.
The DEFINE HEADER statement creates the header definition bar.
The TEXT statement specifies the text to use in the header. _LABEL_ is a dynamic variable that references a value that the data component supplies from the procedure or DATA step, in this example the variable's label.
The first END statement ends the header definition.
The HEADER statement declares bar as the header in the table.
The second END statement ends the column definition.

```plaintext
define column name;
    define header bar;
    text "begin " _label_ " end";
end;
    header=bar;
end;
```

Create the column definition gender. The DEFINE COLUMN statement creates the column definition gender.
The DATANAME= column attribute specifies the name of the column sex in the data component to associate with the column definition gender.
The DEFINE HEADER statement creates the header bar.
The TEXT statement specifies the text Gender to use in the header.
The first END statement ends the header definition.
The HEADER statement declares bar as the header in the table.
The second END statement ends the column definition.

```plaintext
define column gender;
    dataname=sex;
    define header bar;
    text "Gender";
end;
    header=bar;
end;
```

Create three column definitions: age, height, and weight. The three DEFINE COLUMN statements create the column definitions age, weight, and height.
The END statement ends the column definition.

```plaintext
define column age;
end;
define column height;
end;
define column weight;
end;
```

End the table definition. The END statement ends the table definition. The RUN statement executes the TEMPLATE procedure.
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 *SASHELP.*CLASS.*

```sas
data _null_;  
  set sashelp.class;
```

Route the DATA step results to ODS and use the *Checkerboard* table definition. The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. (For more information about using the DATA step with ODS, see Chapter 3, “Output Delivery System and the DATA Step,” on page 39.) The TEMPLATE= suboption tells ODS to use the table definition named *Checkerboard.*

```sas
file print ods=(template='Checkerboard');  
  put _ods_;  
run;
```

Stop the creation of HTML and PDF output. The ODS HTML statement closes the HTML destination and all the files that are associated with it. The ODS PDF statement closes the PDF destination and all the files that are associated with it. You must close the destinations before you can view the output.

```sas
ods html close;  
ods pdf close;
```
Display 10.11  HTML Output (Viewed with Internet Explorer 6.0)

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</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>65.5</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>69.8</td>
<td>112.5</td>
</tr>
<tr>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>64.5</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>59.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Joyce</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>129</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>
The SAS System

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>60</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>51.5</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>Thoruss</td>
<td>M</td>
<td>11</td>
<td>57.3</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>
Overview: ODS Tagsets 551

What Is a Tagset? 551
Why Use the TEMPLATE Procedure to Create Tagsets? 552
Terminology for PROC TEMPLATE 552

Overview: ODS Tagsets

What Is a Tagset?

A tagset is a type of template that defines how to generate a markup language output type from SAS format. You can specify a tagset to create markup language output from ODS. SAS provides tagset definitions for a variety of markup language output. For example, SAS provides several tagsets for XML output, HTML output, XSL,
and more. You can modify any of the SAS tagsets, or you can create your own. By supplying new tagset definitions, you can generate a wider variety of markup language output from SAS output.

### Why Use the TEMPLATE Procedure to Create Tagsets?

The TEMPLATE procedure enables you to customize the look of your SAS output. By using the TEMPLATE procedure, you can modify any of the many markup language tagset definitions that SAS supplies or create a markup language tagset of your own. The Output Delivery System then uses the specified tagset definitions to mark the SAS output which then you can view with an online browser or viewer.

### Terminology for PROC TEMPLATE

For information about terms used in the TEMPLATE procedure, see “Terminology: TEMPLATE Procedure” on page 266

### Markup Language Syntax: TEMPLATE Procedure

```sas
PROC TEMPLATE;
DEFINE TAGSET tagset-path </ STORE=libref.template-store >;
 <tagset-attribute-1; <... tagset-attribute-n;>>
  DEFINE EVENT event-name;
   <event-attribute-1; <...event-attribute-n;>>;
  statements;
  END;
NOTES;
END;
```

### DEFINE TAGSET Statement

**Creates a tagset definition**

**Requirement:** An END statement must be the last statement in the definition.

**Featured in:** All examples found in the “Examples: Creating and Modifying Markup Languages Using the TEMPLATE Procedure” on page 588 section.

```sas
DEFINE TAGSET tagset-path </ STORE=libref.template-store <(READ | WRITE | UPDATE)>>>;
 <tagset-attribute-1; <...tagset-attribute-n;>>
  DEFINE EVENT event-name;
  statements and attributes
  NOTES 'text';
END;
```
Table 11.1  DEFINE TAGSET Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define what is written to the output file.</td>
<td>DEFINE EVENT</td>
</tr>
<tr>
<td>Provide information about the tagset definition.</td>
<td>NOTES</td>
</tr>
<tr>
<td>End a tagset definition, or end the editing of a tagset definition.</td>
<td>END</td>
</tr>
</tbody>
</table>

**Required Arguments**

tagset-path  
specifies where to store the tagset definition.  
**Requirement:** A tagset-path consists of one or more names, separated by periods. Each name represents a directory, or level, in a template store.  
**Default:** PROC TEMPLATE writes the definition to the first template store in the current path where you have write access.  
**Tip:** You can control the item store where the tagset definition is stored by using the ODS PATH statement.  
**Tip:** Names are not case sensitive. However, PROC TEMPLATE uppercases the first letter for easy reading purposes.

**Options**

STORE=libref.template-store  
specifies the template store where the definition is stored in the following form:  

\[
\text{libref.template-store <access-options>}
\]

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, it is created.  
**Interaction:** Using the STORE= option overrides the search list specified in the PATH statement.  
**Restriction:** The STORE= option syntax does not become part of the compiled definition.

access-options  
specifies the access mode for the specified template store, where  

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 will not replace an existing tagset.
UPDATE provides update access as well as read access. If the tagset does not exist, then UPDATE will not create a new tagset. If the tagset does exist, then UPDATE will replace it.

Tagset Attributes

Table 11.2  Tagset Attributes

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the text to use as a copyright.</td>
<td>COPYRIGHT=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify the name of the event to use by default.</td>
<td>DEFAULT_EVENT=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify whether or not the tagset supports embedded stylesheets.</td>
<td>EMBEDDED_STYLESHEET</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Set a numeric value to use as the indentation depth.</td>
<td>INDENT=</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Specify a string which will be printed to the SAS log when the tagset is used.</td>
<td>LOG_NOTE</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify special characters and their translations.</td>
<td>MAP=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify strings to substitute for special characters.</td>
<td>MAPSUB=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Define a nonbreaking space for the markup output.</td>
<td>OUTPUT_TYPE=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Sets a category for the output.</td>
<td>NOBREAKSPACE=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify the tagset definition from which the current definition inherits.</td>
<td>PARENT=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify the text to use as a registered trademark.</td>
<td>REGISTERED_TM=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Define a string to use for line breaks in the markup output.</td>
<td>SPLIT=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify whether or not the tagset allows procedures to place columns one on top of another, or side by side.</td>
<td>STACKED_COLUMNS</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Specify the text to use as a trademark.</td>
<td>TRADEMARK=</td>
<td>HTML, MARKUP</td>
</tr>
</tbody>
</table>

COPYRIGHT= ’(text)’ specifies the text to use as the copyright.

Requirement:  When specifying text, you must 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 definition.

**Requirement:** When specifying an event-name, you must enclose the name of the event in quotation marks.

**Featured in:** Example 3 on page 597

**EMBEDDED_STYLESHEET= YES | ON | NO | OFF**
specifies whether or not the tagset supports embedded stylesheets.

**Default:** The default value is YES or ON which means that embedded stylesheets are supported.

**Tip:** If embedded stylesheets are supported and no stylesheet is specified in the ODS statement, then the stylesheet is written to the top of the output file.

**YES**
supports embedded stylesheets.

**Alias:** ON

**ON**
supports embedded stylesheets.

**Alias:** YES

**NO**
does not support embedded stylesheets.

**Alias:** OFF

**OFF**
does not support embedded stylesheets.

**Alias:** NO

**INDENT=n**
indents output one or more indentation levels, using the number of spaces specified by the INDENT= statement.

**Default:** The default value for XML is 2. For all other ODS destinations, the default value is 0.

**ODS Destinations:** MARKUP

**Featured in:** Example 3 on page 597 and Example 5 on page 601

**n**
specifies a numeric value for the number of spaces that you want the output to indent.

**LOG_NOTE= 'string'**
defines a string that will be printed to the SAS log when the tagset is used.

**string**
specifies the text that is printed to the SAS log.

**Requirement:** You can not specify more than one string at a time.

**MAP= 'characters'**
specifies the special characters that require translation.

**characters**
specifies one or more special characters.

**Requirement:** When listing special characters in the MAP= statement, do not use blank spaces between them.

**Requirement:** When you specify special characters, you must enclose the list of special characters in quotation marks.
**Requirement:** If you use the MAP= statement, you must also use the MAPSUB statement.

**Featured in:** Example 3 on page 597

**MAPSUB= 'strings'**
specifies the text to substitute for the characters that are specified in the MAP= statement.

- **strings**
  Specifies the text strings to substitute for the characters that are specified in the MAP= statement.
  
  **Requirement:** When specifying multiple strings, you must use a forward slash (/) to separate the text strings.
  
  **Requirement:** When specifying strings, you must enclose the entire string list in quotation marks.
  
  **Requirement:** If you use the MAPSUB= statement, you must also use the MAP= statement.

  **Featured in:** Example 3 on page 597

**NOBREAKSPACE= 'string'**
defines a nonbreaking space for the markup output.

- **string**
  specifies the character that is used to define a nonbreaking space.
  
  **Requirement:** When specifying a string, you must enclose the string in quotation marks.
  
  **Restriction:** You can not specify more than one string at a time.

  **Featured in:** Example 3 on page 597

**OUTPUT_TYPE= CSV | HTML | LATEX | WML | XML**
sets a category for the output.

- **CSV**
  produces output with comma-separated values.
  
- **HTML**
  produces hypertext markup language output.
  
- **LATEX**
  produces output in LaTeX, which is a document preparation system for high-quality typesetting.
  
- **WML**
  uses the Wireless Application Protocol (WAP) to produce a wireless markup language.
  
- **XML**
  produces output in extensible markup language.

  **Featured in:** Example 3 on page 597

**PARENT= tagset-path**
specifies the tagset definition from which the current definition inherits.

- **tagset-path**
  specifies the name of a directory in a template store.
  
  **Default:** The current definition inherits from the specified definition in the first template store where you have read access permissions. The PATH statement specifies which locations to search for definitions that were created by PROC TEMPLATE, as well as the order in which to search for them.
**Interaction:** When you specify a parent, all the definition options, attributes, and statements that are specified in the parent’s definition are used in the current definition unless the current definition overrides them.

**Requirement:** When you specify a parent, all of the definition options, attributes, and statements that are specified in the parent’s definition are used in the current definition unless the current definition overrides them.

**Tip:** You can specify a tagset that SAS supplies or a tagset that you defined.

**Tip:** You can control the item store from which the tagset definition is read by using the ODS PATH statement.

**Featured in:** Example 1 on page 588 and Example 9 on page 610

**REGISTERED_TM= \text{(text)}**

specifies the text to use as the registered trademark.

**Requirement:** When specifying \text{text}, you must enclose the text in parentheses and then quotation marks.

**SPLIT= \text{string}**

defines a character string to use for line breaks in the markup output.

**Requirement:** When specifying a string, you must enclose the string in quotation marks.

**Restriction:** You cannot specify more than one string at a time.

**Featured in:** Example 3 on page 597

**STACKED_COLUMNS= YES | ON | NO | OFF**

specifies whether or not the tagset allows procedures to place columns one on top of another, or side by side.

**Default:** The default value is YES or ON, which means that columns are stacked.

**Tip:** To place columns side by side, specify the NO or OFF value.

**Featured in:** Example 3 on page 597 and Example 9 on page 610.

**YES**

stacks columns one on top of another.

**Alias:** ON

**ON**

stacks columns one on top of another.

**Alias:** YES

**NO**

stacks columns side by side each other.

**Alias:** OFF

**OFF**

stacks columns side by side each other.

**Alias:** NO

**TRADEMARK= \text{(text)}**

Specifies the text to use as the trademark.

**Requirement:** When specifying \text{text}, you must enclose the text in parentheses and then quotation marks.
DEFINE EVENT Statement

Defines what is written to the output file

Interaction: Event statement conditions can be added to any DEFINE EVENT statement. For more information about event statement conditions, see “Event Statement Conditions” on page 571

Featured in: Example 6 on page 603 and Example 7 on page 605

DEFINE EVENT event-name;
  <event-attribute-1;...event-attribute-n;>
  BLOCK event-name < / event-condition-statements>;
  BREAK < / event-condition-statements>;
  CLOSE < / event-condition-statements>;
  DELSTREAM stream-name < / event-statement-conditions>;
  FLUSH <event-statement-conditions>;
  NDENT < / event-statement-conditions>;
  OPEN stream-name < / event-statement-conditions>;
  PUT 'text' < / event-statement-conditions>;
  PUTL < / event-statement-conditions>;
  PUTLOG < / event-statement-conditions>;
  PUTQ "text" event-variable < / event-statement-conditions>;
  PUTSTREAM stream-name < / event-statement-conditions>;
  PUTVARS variable-group < / event-statement-conditions>;
  SET $user-defined-event-variable user-defined-value < / event-statement-conditions>;
  TRIGGER event-name <START | FINISH> < / event-statement-conditions>;
  UNBLOCK event-name < / event condition-statements>;
  UNSET $user-defined-event-variable | ALL < / event-statement-conditions>;
  XDENT < / event-statement-conditions>;
  END < / event-statement-conditions>;

Table 11.3 DEFINE EVENT Statements

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set one or more event attributes.</td>
<td>event-attributes</td>
</tr>
<tr>
<td>Disable the specified event.</td>
<td>BLOCK</td>
</tr>
<tr>
<td>Prevent an event from executing.</td>
<td>BREAK</td>
</tr>
<tr>
<td>Close the current stream to which all PUT statement variables are directed</td>
<td>CLOSE</td>
</tr>
<tr>
<td>Delete the specified stream.</td>
<td>DELSTREAM</td>
</tr>
<tr>
<td>Write buffered output to the current output file or stream.</td>
<td>FLUSH</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indent output one more indentation level.</td>
<td>NDENT</td>
</tr>
<tr>
<td>Open or create the specified stream.</td>
<td>OPEN</td>
</tr>
<tr>
<td>Write text or variable data to an output file.</td>
<td>PUT</td>
</tr>
<tr>
<td>Add a new line to the end of the output.</td>
<td>PUTL</td>
</tr>
<tr>
<td>Writes the text, or the value of the event variable to the log.</td>
<td>PUTLOG</td>
</tr>
<tr>
<td>Place quotes around the value in a variable.</td>
<td>PUTQ</td>
</tr>
<tr>
<td>Write the contents of the stream to the current output file.</td>
<td>PUTSTREAM</td>
</tr>
<tr>
<td>Writes the name or value of an event, dynamic, memory, or stream variable to an output file.</td>
<td>PUTVARS</td>
</tr>
<tr>
<td>Specify a user-defined event variable and its value.</td>
<td>SET</td>
</tr>
<tr>
<td>Execute another event.</td>
<td>TRIGGER</td>
</tr>
<tr>
<td>Delete user-defined variables.</td>
<td>UNSET</td>
</tr>
<tr>
<td>Enable a disabled event.</td>
<td>UNBLOCK</td>
</tr>
<tr>
<td>Indent output one less indentation level.</td>
<td>XDENT</td>
</tr>
<tr>
<td>End the definition.</td>
<td>END</td>
</tr>
</tbody>
</table>

### DEFINE Event Statement

Defines what is written to the output file

**DEFINE EVENT** `event-name;`

`<event-attribute-1;...event-attribute-n;>`

### Required Arguments

**event-name**

specifies the name of the event.
Event Attributes

Table 11.4 Event Attributes

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute</th>
<th>Valid destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redirect event output to any of the known types of output that are open.</td>
<td>FILE=</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>Enable the event to use any style element that has been defined.</td>
<td>PURE_STYLE=</td>
<td>MARKUP</td>
</tr>
<tr>
<td>Specify a style element.</td>
<td>STYLE=</td>
<td>HTML, MARKUP</td>
</tr>
</tbody>
</table>

**FILE= BODY | CODE | CONTENTS | FRAME | PAGES | STYLESHEET**
redirects event output to any of the known types of output files that are open.

**Interaction:** The names of the output files correspond to the output file names on the ODS MARKUP statement that are specified with the BODY=, CODE=, CONTENTS=, FRAME=, PAGES=, and STYLESHEET= parameters. For more information about the ODS MARKUP statement, see “ODS MARKUP Statement” on page 109

**ODS Destinations:** HTML, MARKUP
See: For a complete description of the FILE= attribute, see the BODY= option in the ODS MARKUP statement.

**PURE_STYLE= YES | NO**
specifies whether to enable the event to use any style elements that have been defined.

**Default:** NO

**ODS Destinations:** MARKUP
See also: “DEFINE STYLE Statement” on page 288

YES
enables the event to use any style elements that have been defined.

NO
does not enable the event to use any style elements that have been defined.

**STYLE= style-element;**
specifies a style attribute that applies to a particular part of the output.

**ODS Destinations:** HTML, MARKUP
See also: “DEFINE STYLE Statement” on page 288

**Tip:** If you use a carriage return to separate your style attributes, then you must add a space before or after the carriage return to prevent syntax errors. SAS does not interpret a carriage return as a space.

**Featured in:** Example 6 on page 603
**BLOCK Statement**

Disables the specified event

**Tip:** To enable the blocked event, use the UNBLOCK statement.

**Tip:** You can block the same event multiple times, but in order to enable the event, you must use the same number of UNBLOCK statements.

```
BLOCK event-name < / event-statement-conditions>
```

**Required Arguments**

`event-name`

specifies the name of the event.

**Options**

`event-statement-conditions`

specifies event statement conditions that can be added to the BLOCK statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

**BREAK Statement**

Stops an event from executing

**Tip:** The BREAK statement is most useful when combined with event conditions.

```
BREAK < / event-statement-conditions>
```

**Options**

`event-statement-conditions`

specifies event statement conditions that can be added to the BREAK statement.

See: For information about these options, see “Event Statement Conditions” on page 571.
CLOSE Statement

Closes the current stream and directs all future output to the output file

`CLOSE < /event-statement-conditions>;`

Options

`event-statement-conditions` specifies event statement conditions that can be added to the CLOSE statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

DELSTREAM Statement

Deletes the specified stream

`DELSTREAM stream-name < /event-statement-conditions>;`

Required Arguments

`stream-name` specifies the name of the stream.

Options

`event-statement-conditions` specifies event statement conditions that can be added to the DELSTREAM statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

FLUSH Statement

Writes any buffered output to the current output file or stream

`FLUSH</event-statement-conditions>;`
Options

`event-statement-conditions` specifies event statement conditions that can be added to the FLUSH statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

---

**NDENT Statement**

Indents output one more indentation level than the number of spaces specified by the INDENT= statement.

Interaction: The start position of the indentation level is set by the INDENT= attribute.

Featured in: Example 3 on page 597 and Example 5 on page 601

```
NDENT < / event-statement-conditions >;
```

---

Options

`event-statement-conditions` specifies event statement conditions that can be added to the NDENT statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

---

**OPEN Statement**

Opens the specified stream or creates one if the specified stream does not exist.

Interaction: If another stream is open, then it will be closed when you specify a new stream to be opened.

Interaction: All text or variable data specified in the PUT statements that occur after the OPEN statement, will append to the stream instead of the output file.

```
OPEN stream-name < / event-statement-conditions >;
```

---

**Required Arguments**

`stream-name` specifies the name of the stream.
PUT Statement

Writes the text, or the value of an event variable to an output file

Required Argument

text

specifies a text string that provides information about your output.

Interaction: The PUT statement pairs strings with variables. If a string is followed by a variable, 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 will be output.

Requirement: The text must be enclosed in quotation marks.

VALUE

specifies the value of the event variable.

Interaction: The PUT statement pairs strings with variables. If a string is followed by a variable, 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 will be output.

See: For a list of event variables, see “List of Event Variables” on page 572

Options

event-statement-conditions

specifies event statement conditions that can be added to the PUT statement.

See: For information about these options, see “Event Statement Conditions” on page 571.
PUTL Statement

Adds a new line to the end of the output

**Alias:** CR, NL, or LF

**Tip:** Use the PUTL statement when your event output is large.

```
PUTL </event-statement-conditions>;
```

**Options**

`event-statement-conditions`

specifies event statement conditions that can be added to the PUTL statement.

**See:** For information about these options, see “Event Statement Conditions” on page 571.

PUTLOG Statement

Writes the text, or the value of the event variable to the log

**Requirement:** You must enclose the text string in quotation marks.

```
PUTLOG 'text' </event-statement-conditions>;
```

**Required Argument**

`text`

specifies a text string that provides information about your output.

**Interaction:** The PUTLOG statement pairs strings with variables. If a string is followed by a variable, they become a pair. If the variable has a value, then the pair becomes output. If the variable does not have value, then neither will be output.

**Requirement:** The `text` must be enclosed in quotation marks.

**Options**

`event-statement-conditions`

specifies event statement conditions that can be added to the PUTLOG statement.

**See:** For information about these options, see “Event Statement Conditions” on page 571.
PUTQ Statement

Places quotes around the value in an event variable or a style variable

Featured in: Example 7 on page 605

PUTQ 'text' event-variable < / event-statement-conditions >;

Required Argument

text

specifies a text string.

Requirement: The text must be enclosed in quotation marks.

Interaction: The PUTQ statement pairs strings with variables. If a string is followed by a variable, 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 will be output.

event-variable

specifies the event variable.

See: “List of Event Variables” on page 572.

Options

event-statement-conditions

specifies event statement conditions that can be added to the PUTQ statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

PUTSTREAM Statement

Writes the contents of the stream to the current output file

PUTSTREAM stream-name < / event-statement-conditions >;

Required Arguments

stream-name

specifies the name of the stream.
Options

**event-statement-conditions**
specifies event statement conditions that can be added to the PUTSTREAM statement.

**See:** For information about these options, see “Event Statement Conditions” on page 571.

---

**PUTVARS Statement**

Writes the name or value of an event, dynamic, memory, or stream variable to an output file

**Interaction:** The PUTVARS statement loops through all the variables in the variable group. Each iteration populates special variables which can be used in the format.

```
PUTVARS variable-group <{/event-statement-conditions}>
```

---

**Required Argument**

**variable-group**
specifies the variables to use in each iteration when you specify the name or value in the variable.

**Interaction:** The PUTVAR statement pairs strings with variables. If a string is followed by a variable, 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 will be output.

**EVENT**
specifies the name of an event variable.

**See:** “List of Event Variables” on page 572

**STYLE**
specifies the style elements available from the current ODS style definition being used.

**DYNAMIC**
specifies the name of a dynamic variable.

**MEMORY**
specifies the name of the variable created in the SET statement of the DEFINE EVENT statement.

**Requirement:** Memory variables must be preceded by the '$' symbol.

**STREAM**
specifies the name of the variable stream.

**Requirement:** Stream variables must be preceded by the '$$' symbol.
Options

*event-statement-conditions*

specifies event statement conditions that can be added to the PUTVARS statement.

**See:** For information about these options, see “Event Statement Conditions” on page 571.

---

**SET Statement**

Specifies a user-defined variable and its value

**Requirement:** The user-defined variable must be preceded by a ’$’ character.

**Tip:** User-defined variables are case insensitive.

```
SET $user-defined-event-variable user-defined-value </ event-statement-conditions>
```

**Required Arguments**

*user-defined-event-variable*

specifies the name of the variable that you want to create.

**Requirement:** The *user-defined-event-variable* must be preceded by a ’$’ character.

**Tip:** User-defined-event-variables are case insensitive.

*user-defined-value*

specifies the value of the *user-defined-variable*.

**Tip:** Any value can be used for the *user-defined-variable*. You can assign an existing user-defined-variable name as a value for the variable.

**Options**

*event-statement-conditions*

specifies event statement conditions that can be added to the SET statement.

**See:** For information about these options, see “Event Statement Conditions” on page 571.

---

**TRIGGER Statement**

Executes another event

**Tip:** The TRIGGER statement explicitly requests a specific action of an event.

**Featured in:** Example 3 on page 597, Example 4 on page 599, Example 5 on page 601, and Example 6 on page 603
TRIGGER event-name <START | FINISH> <! event-statement-conditions>;

Required Arguments

event-name
  specifies the name of the event.

Without Options

If a triggered event does not have start or finish sections, then it will run the statements it does have.

Options

START
  specifies the start section of an event.
  Interaction: If you are in the start section of an event, then any event triggered will also run its start section.

FINISH
  specifies the finish section of an event.
  Interaction: If you are in the finish section of an event, then any event triggered will also run its finish section.

event-statement-conditions
  specifies event statement conditions that can be added to the TRIGGER statement.
  See: For information about these options, see “Event Statement Conditions” on page 571.

UNBLOCK Statement

Enables a disabled event

Interaction: To disable an event, use the BLOCK statement.

Requirement: Because you can block the same event multiple times, in order to enable the event, you must use the same number of UNBLOCK statements as BLOCK statements.

UNBLOCK event-name <! event-statement-conditions>;

Required Arguments

event-name
  specifies the name of the event.
Options

`event-statement-conditions`

specifies event statement conditions that can be added to the UNBLOCK statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

---

**UNSET Statement**

Deletes the user-defined-variables and their values

**Requirement:** To specify the `user-defined-variable`, you must precede the name with a ')' character.

---

**UNSET**

$`user-defined-event-variable` | **ALL**

`< event-statement-conditions >`

---

**Required Arguments**

$`user-defined-event-variable`

specifies the name of the variable that you want to delete.

**Requirement:** The `user-defined-event-variable` must be preceded by a ')' character and no space.

**Tip:** User-defined-event-variables are case insensitive.

**ALL**

deletes all `user-defined-event-variables`.

---

**Options**

`event-statement-conditions`

specifies event statement conditions that can be added to the UNSET statement.

See: For information about these options, see “Event Statement Conditions” on page 571.

---

**XDENT Statement**

Indents output one less indentation level, using the number of spaces specified by the **INDENT** attribute

**Interaction:** The starting level of indentation is set by the **INDENT** statement.

**Featured in:** Example 3 on page 597 and Example 5 on page 601

---

**XDENT**

`< event-statement-conditions >`;
Options

*event-statement-conditions*

specifies event statement conditions that can be added to the XDENT statement.

**See:** For information about these options, see “Event Statement Conditions” on page 571.

---

**END Statement**

Ends the event definition

END;

---

**Event Statement Conditions**

Conditions can be added to any DEFINE EVENT statement. A condition must be preceded with a slash (/).

*event-statement < / event-statement-condition >;

---

**Event Statement Conditions**

*event-statement*

specifies any of the DEFINE EVENT statements.

*event-statement-condition*

specifies the type of condition.

Values for the *event-condition-statements* are one of the following:

**ANY**

checks a list of variables for values. If any of the variables has a value, then the condition is true and the statement executes.

For example:

```
put “One of our variables has a value!”
   nl/if any(background, foreground, cellpadding, cellspacing);
```

**CMP**

compares, for equality, a string to a variable or list of variables.

For example:

```
put “The foreground is blue!” nl/if cmp(“blue”,foreground);
```
CONTAINS  
searches the first argument for the second argument.  
For example:  

```plaintext
set $junk "some random text";
put "junk contains 'ran'" nl/if contains($junk, "ran");
```

EXIST | EXISTS  
checks a variable, or a list of variables, to determine if a value exists. If all of the variables have a value, then the condition is true and the statement executes. If a variable has an empty string of length 0, then the value does not exist and the statement does not execute.  
For example:  

```plaintext
put "All of our variables have a value!"
   nl/if exists(background, foreground, cellpadding, cellspacing);
```

**Tip:** Use the MISSING event variable to determine if a value is missing.

IF | WHEN  
tests for existence or equality. IF and WHEN are optional and interchangeable. An IF or WHEN condition compares values and strings or checks variables for values.  
For example, all of the following are equivalent:  

```plaintext
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);
```

NOT  
negates a condition. You can use the keyword NOT or the characters '!' or '^'.  
For example:  

```plaintext
put "The foreground is not red!" nl/if !cmp("red", foreground);
```

---

**Event Variables**

---

**List of Event Variables**

Event variables include text, formatting, and data values. These variables originate from many places, such as the table definition, the procedure, title, byline processing, and more. Event variables also include any style attributes that you are using in your program. The following table lists the event variables that are used in the DEFINE EVENT statement of PROC TEMPLATE.
### Table 11.5  Event Variables

<table>
<thead>
<tr>
<th>Event Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| ABBR           | specifies an abbreviation for the event variable.  
| ACRONYM        | specifies an acronym for the event variable.  
| ALT            | specifies an alternate description of the event variable.  
| AFTER          | specifies that the current note is an after note.  
| ANCHOR         | specifies the current anchor, which is the last value of the anchor tag (for example, IDX).  
| ARCHIVE        | used by the SAS/GRAPH to specify the Java archive (.jar) file to be used. CODEBASE must be used to specify the directory containing the .jar file.  
| ASIS           | specifies how to handle leading spaces and line breaks.  
| ATTR_NAME      | used by the DATA step interface.  
| ATTR_VALUE     | used by the DATA step interface.  
| AUTHOR         | specifies the author of the output. Set from the ODS statement, or, by default, is the user that is running SAS.  
| BACKGROUND     | specifies the color of the background.  
| BACKGROUNDIMAGE| specifies the background image. This image will be stretched.  
| BEFORE         | specifies that the current note is a before note.  
| BASENAME       | specifies the name of the BASE= option as set in the ODS statement.  
| BODY_NAME      | specifies the name of the body file.  
| BODYSIZE       | specifies the width of the frame in the HTML frame file that displays the body file.  
| BODY_TITLE     | specifies the title of the body file.  
| BODY_URL       | specifies the URL of the body file.  
| BORDERCOLOR    | specifies the color of the border if the border is only one color.  
<p>| BORDERCOLORDARK| specifies the darker color in the border that uses two colors to create a three-dimensional effect.  |</p>
<table>
<thead>
<tr>
<th>Event variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORDERCOLORLIGHT</td>
<td>specifies the lighter color in the border that uses two colors to create a three-dimensional effect.</td>
</tr>
<tr>
<td>BORDERWIDTH</td>
<td>specifies the width of the border of the table.</td>
</tr>
<tr>
<td>BOTTOMMARGIN</td>
<td>specifies the bottom margin for the document.</td>
</tr>
<tr>
<td>BULLET</td>
<td>specifies the string to use for bullets in the contents file.</td>
</tr>
<tr>
<td>CAPTION</td>
<td>specifies the caption for the table.</td>
</tr>
<tr>
<td>CELLHEIGHT</td>
<td>specifies the height of the cell.</td>
</tr>
<tr>
<td>CELLPADDING</td>
<td>specifies the amount of white space on each of the four sides of the text in a cell.</td>
</tr>
<tr>
<td>CELLSSpacing</td>
<td>specifies the width of the spacing between cells.</td>
</tr>
<tr>
<td>CELLWIDTH</td>
<td>specifies the width of the cell.</td>
</tr>
<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 definition.</td>
</tr>
<tr>
<td>CLASSID</td>
<td>used by SAS/GRAPH to specify where to place the ActiveX files in the Windows registry.</td>
</tr>
<tr>
<td>CODE</td>
<td>used by SAS/GRAPH to specify which Java class to activate when the applet opens.</td>
</tr>
<tr>
<td>CODEBASE</td>
<td>used by SAS/GRAPH to specify the directory where the Java archive (.JAR) files are located. ARCHIVE must also be used to specify the .JAR file. For ActiveX, the location of the ActiveX set-up file is specified.</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>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>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>COLSPAN</td>
<td>specifies the number of columns that the cell spans.</td>
</tr>
<tr>
<td>Event variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>COLSTART</td>
<td>specifies the column number where the cell starts.</td>
</tr>
<tr>
<td>CONTENTS_NAME</td>
<td>specifies the name of the contents file.</td>
</tr>
<tr>
<td>CONTENTPOSITION</td>
<td>specifies the position, within the frame file, of the frames that display the contents and the page files.</td>
</tr>
<tr>
<td>CONTENTSCROLLBAR</td>
<td>specifies whether to put a scroll bar in the frames that display the contents and the page files.</td>
</tr>
<tr>
<td>CONTENTSIZE</td>
<td>specifies the width of the frames that display the contents and the page files.</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>CONTRASTCOLOR</td>
<td>specifies alternate colors for maps. The alternate colors are applied to the blocks on region areas in block maps.</td>
</tr>
<tr>
<td>COORDINATE</td>
<td>used by SAS/GRAPH to specify the coordinates for a specified shape.</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>DATA_ROW</td>
<td>specifies that the current row is a data row.</td>
</tr>
<tr>
<td>DATE</td>
<td>specifies the date.</td>
</tr>
<tr>
<td>DEFAULT_JUST</td>
<td>specifies the default horizontal justification. Internal use only.</td>
</tr>
<tr>
<td>DEFAULT_VJUST</td>
<td>specifies the default vertical justification. Internal use only.</td>
</tr>
<tr>
<td>DEST_FILE</td>
<td>specifies the current destination file: body, contents, pages, frame, code, or stylesheet.</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 set with the DATANAME= attribute in the column definition.</td>
</tr>
<tr>
<td>DROPSHADOW</td>
<td>specifies a drop shadow effect for text in a graph.</td>
</tr>
<tr>
<td>EMPTY</td>
<td>sets a flag to determine whether an event is called as an empty tag.</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>ENDCOLOR</td>
<td>specifies the end color for a gradient effect in a graph.</td>
</tr>
<tr>
<td>Event variable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EVENT_NAME</td>
<td>specifies the requested event name.</td>
</tr>
<tr>
<td>FILLRULEWIDTH</td>
<td>specifies the width of the fill rule.</td>
</tr>
<tr>
<td>FIRSTPAGE</td>
<td>specifies the first page of the output file.</td>
</tr>
<tr>
<td>FLYOVER</td>
<td>specifies the text to show in a tool tip for the cell.</td>
</tr>
<tr>
<td>FONT</td>
<td>specifies the font definition.</td>
</tr>
<tr>
<td>FONT_FACE</td>
<td>specifies the name of the font face.</td>
</tr>
<tr>
<td>FONT_SIZE</td>
<td>specifies the size of the font.</td>
</tr>
<tr>
<td>FONT_STYLE</td>
<td>specifies the style of the font.</td>
</tr>
<tr>
<td>FONT_UNDERLINE</td>
<td>specifies the underline character.</td>
</tr>
<tr>
<td></td>
<td>FONT_UNDERLINE is only used by the ODS MARKUP statement.</td>
</tr>
<tr>
<td>FONT_WEIGHT</td>
<td>specifies the weight of the font.</td>
</tr>
<tr>
<td>FONT_WIDTH</td>
<td>specifies the width of the font.</td>
</tr>
<tr>
<td>FOREGROUND</td>
<td>specifies the color of the foreground.</td>
</tr>
<tr>
<td>FRAME</td>
<td>specifies the type of frame to use on a table.</td>
</tr>
<tr>
<td>FRAMEBORDER</td>
<td>specifies whether to put a border around the frame for an HTML file that uses frames.</td>
</tr>
<tr>
<td>FRAMEBORDERWIDTH</td>
<td>specifies the width of the border around the frames for an HTML file that uses frames.</td>
</tr>
<tr>
<td>FRAME_NAME</td>
<td>specifies the name of the frame file.</td>
</tr>
<tr>
<td>FRAMESPACING</td>
<td>specifies the width of the space between frames for an HTML file that uses frames.</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 given in the ODS PATH statement.</td>
</tr>
<tr>
<td>GRAPH_PATH_URL</td>
<td>specifies the URL of the graph.</td>
</tr>
<tr>
<td>GRADIENT_DIRECTION</td>
<td>specifies the direction of the gradient effect in either the X or Y axis direction to influence the graph background, legend background, charts, walls, floors, etc.</td>
</tr>
<tr>
<td>HIDDEN</td>
<td>specifies that the current object is hidden.</td>
</tr>
<tr>
<td>HREFTARGET</td>
<td>specifies the window or frame in which to open the target of the link.</td>
</tr>
<tr>
<td>HTMLCLASS</td>
<td>specifies the name of the stylesheet class to use for the table or cell.</td>
</tr>
<tr>
<td>HTMLCONTENTTTYPE</td>
<td>specifies the value of the content type for pages that you send directly to a web server rather than to a file.</td>
</tr>
<tr>
<td>Event variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HTMLDOCTYPE</td>
<td>specifies the entire doctype declaration for the HTML document.</td>
</tr>
<tr>
<td>HTMLID</td>
<td>specifies the ID for the table or cell.</td>
</tr>
<tr>
<td>HTMLSTYLE</td>
<td>specifies individual attributes and values for the table or cell.</td>
</tr>
<tr>
<td>IMAGE</td>
<td>specifies the image to appear in the background. This image can be positioned or tiled.</td>
</tr>
<tr>
<td>IN_ASSOCIATION</td>
<td>specifies the combination of a caption and a table. Associations are used in PROC REPORT, PROC TABULATE, and PROC FREQ cross-tabulations.</td>
</tr>
<tr>
<td>IN_CAPTION</td>
<td>specifies a caption.</td>
</tr>
<tr>
<td>IS_NOTE</td>
<td>specifies a note.</td>
</tr>
<tr>
<td>IS_STACKED</td>
<td>specifies that the columns are stacked.</td>
</tr>
<tr>
<td>IS_TITLE</td>
<td>specifies that the current procedure title remains a title.</td>
</tr>
<tr>
<td>JUST</td>
<td>specifies the horizontal justification.</td>
</tr>
<tr>
<td>LABEL</td>
<td>specifies the label for the variable. Set with the LABEL= attribute in the column definition.</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>specifies the language of the current output. LANGUAGE is set when it is only an Asian language.</td>
</tr>
<tr>
<td>LEFTMARGIN</td>
<td>specifies the left margin for the document.</td>
</tr>
<tr>
<td>LINESTYLE</td>
<td>specifies the line type to use in a graph. You can use SAS/GRAPH line types 1–46.</td>
</tr>
<tr>
<td>LINKCOLOR</td>
<td>specifies the color for links that have not yet been visited.</td>
</tr>
<tr>
<td>LISTENTRYANCHOR</td>
<td>specifies whether to make the entry in the table of contents a link to the body file.</td>
</tr>
<tr>
<td>LONGDESC</td>
<td>specifies the long description of an event variable.*</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 can be used to represent different categories of missing data by assigning the letters A – Z or an underscore.</td>
</tr>
<tr>
<td>NAME</td>
<td>specifies the name of the variable. NAME is set with the VARNAME= attribute in the column definition.</td>
</tr>
<tr>
<td>Event variable</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><em>NAME</em></td>
<td>contains the name of the current variable.</td>
</tr>
<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>NOBREAKSPACE</td>
<td>specifies how to handle spaces at line breaks.</td>
</tr>
<tr>
<td>NO_WRAP</td>
<td>specifies that the current cell should not wrap text or insert hyphens.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>specifies the operator. OPERATOR is set from the ODS statement, or, by default, it is the user that is running SAS.</td>
</tr>
<tr>
<td>OUTPUTHEIGHT</td>
<td>specifies the height for a graph or the graphics in the output.</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>OUTPUTWIDTH</td>
<td>specifies the width of a table, graph, or line thickness.</td>
</tr>
<tr>
<td>OVERHANGFACTOR</td>
<td>specifies the upper limit for extending the width of the column.</td>
</tr>
<tr>
<td>PAGEBREAKHTML</td>
<td>specifies the HTML to place at page breaks.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>specifies the page count since the files were opened.</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 the ODS statement.</td>
</tr>
<tr>
<td>PATH_NAME</td>
<td>specifies the path name.</td>
</tr>
<tr>
<td>PATH_URL</td>
<td>specifies the path location.</td>
</tr>
<tr>
<td>POSTHTML</td>
<td>specifies the HTML code to place after the table or cell.</td>
</tr>
<tr>
<td>POSTIMAGE</td>
<td>specifies the image to place after the table or cell.</td>
</tr>
<tr>
<td>POSTTEXT</td>
<td>specifies the text to place after the table or cell.</td>
</tr>
<tr>
<td>PRECISION</td>
<td>specifies the number of places to the right of the decimal. PRECISION is used by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>PREHTML</td>
<td>specifies the HTML code to place before the table or cell.</td>
</tr>
<tr>
<td>Event variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PREIMAGE</td>
<td>specifies the image to place before the table or cell.</td>
</tr>
<tr>
<td>PRETEXT</td>
<td>specifies the text to place before the table or cell.</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>PROTECTSPECIALCHARACTERS</td>
<td>specifies how the less-than (&lt;) and greater-than (&gt;) signs and the ampersand (&amp;) are interpreted.</td>
</tr>
<tr>
<td>RAWVALUE</td>
<td>specifies the base64 encoding of the stored machine representation of the original value.</td>
</tr>
<tr>
<td>REF_ID</td>
<td>specifies the reference ID for references to columns. Used by the XML LIBNAME engine for the OMDBM format type.</td>
</tr>
<tr>
<td>RIGHTMARGIN</td>
<td>specifies the right margin for the document.</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>RULES</td>
<td>specifies the type of line that is used between table cells.</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>SCALE</td>
<td>specifies the total number of places in the floating point number. SCALE is used by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>SECTION</td>
<td>specifies the head, body, or foot of the table.</td>
</tr>
<tr>
<td>SHAPE</td>
<td>is used with SAS/GRAPH to specify the type of shape to draw.</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>STARTCOLOR</td>
<td>specifies the start color for a gradient effect in a graph.</td>
</tr>
<tr>
<td>STATE</td>
<td>specifies the current state of the event, which is either START or FINISH.</td>
</tr>
<tr>
<td>STYLE</td>
<td>specifies the current style that is being used.</td>
</tr>
<tr>
<td>STYLESHEET_NAME</td>
<td>specifies the name of the stylesheet file.</td>
</tr>
<tr>
<td>Event variable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STYLESHEET_TITLE</td>
<td>specifies the title of the stylesheet file.</td>
</tr>
<tr>
<td>STYLESHEET_URL</td>
<td>specifies the URL of the stylesheet file.</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>specifies a summary of the table.</td>
</tr>
<tr>
<td>TAGATTR</td>
<td>specifies the text to insert in the HTML.</td>
</tr>
<tr>
<td>TAG_NAME</td>
<td>specifies the tag name.</td>
</tr>
<tr>
<td>TAGSET</td>
<td>specifies the name of the current tagset.</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>TARGET</td>
<td>specifies the target that is associated with the URL.</td>
</tr>
<tr>
<td>TEXT</td>
<td>specifies the tag names. TEXT is used by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>TIME</td>
<td>specifies the time.</td>
</tr>
<tr>
<td>TITLE</td>
<td>specifies the title from the ODS statement.</td>
</tr>
<tr>
<td>TOCLEVEL</td>
<td>specifies the table of contents level.</td>
</tr>
<tr>
<td>TOPMARGIN</td>
<td>specifies the top margin for the document.</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 how many procedures that have run since the ODS was opened.</td>
</tr>
<tr>
<td>TRANSPARENCY</td>
<td>specifies the level of transparency for a graph.</td>
</tr>
<tr>
<td>TRANTAB</td>
<td>specifies the translation table name for character conversions.</td>
</tr>
<tr>
<td>TRIGGER_NAME</td>
<td>specifies the name of the event that is triggered.</td>
</tr>
<tr>
<td>TYPE</td>
<td>specifies the STRING, DOUBLE, CHAR, BOOL, or INT data type.</td>
</tr>
<tr>
<td>URL</td>
<td>specifies the URL to link to when the item is clicked.</td>
</tr>
<tr>
<td>VALUE</td>
<td>specifies the current value.</td>
</tr>
<tr>
<td><em>VALUE</em></td>
<td>contains the value of the current variable.</td>
</tr>
<tr>
<td>VALUECOUNT</td>
<td>specifies the count of the variable.</td>
</tr>
<tr>
<td>VISITEDLINKCOLOR</td>
<td>specifies the color for links that have been visited.</td>
</tr>
<tr>
<td>VJUST</td>
<td>specifies the vertical justification.</td>
</tr>
<tr>
<td>WATERMARK</td>
<td>specifies whether to make the image that is specified by BACKGROUNDIMAGE into a watermark.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>specifies the width. Most commonly used for COLSPECS.</td>
</tr>
</tbody>
</table>
### Event variable Description

<table>
<thead>
<tr>
<th>Event variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLDATAFORM</td>
<td>specifies whether the tag for an element to contain SAS variable information (name and data) is to appear in an open element or an enclosed attribute format. XMLDATAFORM is used by the XML LIBNAME engine.</td>
</tr>
<tr>
<td>XMLMETADATA</td>
<td>specifies the metadata for the XML tagset.</td>
</tr>
<tr>
<td>XMLSCHEMA</td>
<td>specifies whether or not to generate schema-related information. XMLSCHEMA is used by the XML LIBNAME engine.</td>
</tr>
</tbody>
</table>

* SAS includes these accessibility and compatibility features to improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology that are adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

### NOTES Statement

Provides information about the tagset definition

**Tip:** The NOTES statement becomes part of the compiled tagset definition, which you can view with the SOURCE statement.

**Featured in:** Example 3 on page 597 and Example 9 on page 610

**NOTES 'text';**

### Required Arguments

**text**

provides information about the tagset.

### END Statement

Ends the tagset definition

**END;**
Getting Familiar with Tagsets

Listing Tagset Names

SAS provides a set of tagset definitions. To get a list of the tagset names that SAS supplies, plus any tagsets that you created and stored in the SASHELP.TMPLMST template store, submit the following SAS statements:

```sas
proc template;
  list tagsets;
run;
```

By default, PROC TEMPLATE lists the tagsets in SASHELP.TMPLMST and SASUSER.TEMPLAT. Typically, you have read-only access 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.

Note: The tagset names that begin with SAS are used by the XML LIBNAME engine and are supported by SAS. For example, TAGSETS.SASXMOG and TAGSETS.SASXMOIM are fully supported by SAS.

Specifying Tagset Names

To specify a SAS tagset stored in SASHELP.TMPLMST or a tagset that you 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 created and stored in an item store other than SASUSER.TEMPLAT, assign the item store to the ODS search path with the ODS PATH statement. For information about the ODS PATH statement, see “ODS PATH Statement” on page 149.

Viewing the Contents of a Tagset Definition

To view the contents of a tagset definition, you can 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 on `Tagsets` to view the contents of that item store or directory.
  4. Double-click on the tagset definition that you wish to view. For example, the CHTML tagset definition 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 tagset definitions that SAS provides, double-click the Tagset item store.

4. Right-click the tagset definition, such as Rtf, and select Open. The tagset definition is displayed in the Template Browser window.

TEMPLATE Procedure

1. To see the source for a tagset definition, 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.ctml;
   
   If you look at the source for TAGSETS.CHTML, you see that it consists of:
   □ a DEFINE TAGSET statement that names the tagset definition
   □ event definitions that define what is written to the output file
   □ tagset definition attributes, such as output type and the character to use for line breaks.

What Are Events?

A tagset definition 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 your own tagset definition. In addition, you can define your own events.

□ The DEFINE EVENT statement assigns a name to an event definition

□ An event definition can include start and/or finish sections that specify different actions. If the event definition does not include either a start or finish section, the event is stateless, which means that no matter how the event is called, all of the actions in the event are executed. If an event has a finish section, a start section is assumed if there are statements above the finish section.

□ An event definition can execute another event using the TRIGGER statement. If you are in the start section of an event, then any event triggered will also run its start section. If you are in the finish section, then the triggered event will run its finish section. If a triggered event does not have start or finish sections, then it will run the statements that it does have. A trigger can also explicitly ask for an event’s specific section. See Example 4 on page 599

□ Events can perform actions based on conditions.

□ For the most part, an event consists of PUT statements, text, and event variables.
For example, here is a simple event definition for an HTML table output.

```sas
define event table;
start:
  put "<table>" nl;
finish:
  put "</table>" nl;
end;
```

In the event definition:

1. The DEFINE EVENT statement begins the event and assigns it the name TABLE.
2. The START section defines the beginning portion for the event, and the FINISH section defines the ending portion of the event. An event definition for a table needs START and FINISH sections because ODS needs to know how to define the beginning and how to define the ending. ODS will also expect 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, along with a new line after each tag.

The following event definition does not include a start and finish section, and 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, enclosed with the <TD> and </TD> tags.

```sas
define event data;
  put "<TD>";
  put VALUE;
  put "</TD>";
end;
```

**What Are Variables?**

A variable is a programming structure used to hold data. A variable holds the data assigned to it until a new value is assigned or the program is finished. Each variable has a unique name and holds information that either is internal information to handle the requested output (metadata used by ODS or the XML LIBNAME engine) or 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 used by tagsets can be divided into two groups: internally generated and user created.

There are 3 logical divisions of internally generated variables:

- **event variables** include text, formatting, and data values. These variables can originate in many places such as the table definition, the procedure, title, or byline processing.
- **style variables** are specified by the ODS style attributes currently in use. The style variables are only differentiated from other event variables in that you know exactly where they originate.
- **dynamic variables** are dynamically created within SAS. Because they are dynamically created, their names, or how they are used, is unknown. These variables are dynamic because they are not defined by ODS but rather the variables are defined by applications such as SAS/GRAPH and the XML LIBNAME engine. Dynamic variables are designated by a preceding @ symbol. Dynamic variables can be listed with the “DYNAMIC Statement” on page 425. For more information about SAS/GRAPH, see SAS/GRAPH Reference, Volumes 1 and 2.
There are two types of user-created variables:

- **memory variables** are created with the SET statement, within the DEFINE EVENT statement. Once created, memory variables are globally available in all events. They persist until they are deleted. Memory variables are designated by a preceding ‘$’ symbol.

- **stream variables** are different from memory variables in that they can hold very large amounts of data. They can hold very large amounts of data because as they grow, they are written to disk as needed. Opening a stream variable redirects all output from the put statements to the stream, until it is closed. Stream variables can also be opened, closed, flushed, set and unset.

### Displaying Event Variables and Their Values

Because variables represent data, their values may or may not be present, depending on the SAS procedure and the job. For example, some variables have values only if you specified them with procedure options or style options. Other variables have values because the internal information is needed, such as how many columns are in the output. For example, TAGSETS.CHTML contains the event definition COLSPECS, which uses the event variable COLCOUNT so that ODS knows how many columns are in the output:

```sas
define event colspecs;
   put "<p>" nl "<table";
   putq " columns=" COLCOUNT;
   put " cellpadding=2 border=1" nl;
end;
```

To determine which variables have values and what the values are, submit your SAS program using the EVENT_MAP statement. For more information, see “Defining a Tagset Using the EVENT_MAP Tagset” on page 586. For a list of event variables and their descriptions, see “List of Event Variables” on page 572.

### Creating Your Own Tagsets

#### Methods for Creating Your Own Tagsets

To create a tagset, you use the TEMPLATE procedure to define the tagset definition. In general, there are three methods that you can use to create your own tagset.

- Define a tagset definition through inheritance.
- Copy an existing tagset definition, then modify it.
- Define your own tagset definition.

#### Inheriting Events in a Tagset Definition

Tagsets can inherit events from each other. For example, the SAS tagset TAGSETS.WMLOLIST inherits most of its events from TAGSETS.WML, and
Creating Your Own Tagsets

Chapter 11

TAGSETS.IMODE gets most of its events from TAGSETS.CHTML. Inheriting events from an existing tagset definition is the easiest way to define a new tagset definition.

To inherit events, a tagset definition 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 definition, all of the tagset options, attributes, and statements that are specified in the parent's definition are used in the new definition unless the new definition overrides them. That is, in the new tagset definition, an event can override the operation of the same-named event defined in the parent tagset. For example, if the parent tagset defines an event named TABLE, you can change the operation in the new tagset by redefining the event named TABLE.

For an example of inheriting events in a tagset definition, see Example 1 on page 588.

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, you can create your own tagset with customized events. However, in order to customize the events, you must know the names of the events that ODS uses.

A good way to start defining your customized tagset is to use the EVENT_MAP tagset that SAS supplies in order to determine which events are triggered and which variables are used by an event to send output from a SAS process to an output file. When you run a SAS process with TAGSETS.EVENT_MAP, ODS writes XML markup to an output file that shows all event names and variable names as tags. In the output, tag names are the event names. Tag attributes are the variables that have values for those events.

For example, the following statements run ODS MARKUP with TYPE=EVENT_MAP to see which events and variables ODS uses for various parts of the PROC PRINT output:

```ods markup type=event_map file='custom-tagset-filename.xml';
proc print data=sashelp.class;
  where Height gt 60;
run;
ods markup close;
```

Here is the listing output and resulting XML file:

Output 11.1  Listing Output

<table>
<thead>
<tr>
<th>The SAS System</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
<td>Name</td>
</tr>
<tr>
<td>1</td>
<td>Alfred</td>
</tr>
<tr>
<td>3</td>
<td>Barbara</td>
</tr>
<tr>
<td>4</td>
<td>Carol</td>
</tr>
<tr>
<td>5</td>
<td>Henry</td>
</tr>
<tr>
<td>8</td>
<td>Janet</td>
</tr>
<tr>
<td>9</td>
<td>Jeffrey</td>
</tr>
<tr>
<td>12</td>
<td>Judy</td>
</tr>
<tr>
<td>14</td>
<td>Mary</td>
</tr>
<tr>
<td>15</td>
<td>Philip</td>
</tr>
<tr>
<td>16</td>
<td>Robert</td>
</tr>
<tr>
<td>17</td>
<td>Ronald</td>
</tr>
<tr>
<td>19</td>
<td>William</td>
</tr>
</tbody>
</table>
For example, in the XML output that is generated by EVENT_MAP, you can see that PROC PRINT uses events named DOC_HEAD, PROC, TABLE, and so on. The TABLE event uses data from event variables like STATE, CLASS, and TYPE. Once you know
the events and variables that are used to generate the output, then you can define your own tagset definition and customize the events. For example, you could redefine the TABLE event to produce your own output.

To define a tagset to customize your own output, you could start by specifying TAGSETS.EVENT_MAP as the parent tagset. Then, as you redefine events to customize output, they will replace the default events 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 your tagset definition. When you’re satisfied with the customized output, remove the EVENT_MAP inheritance and the empty events. Then, your output will reflect only the events you defined.

Note: When you first run a SAS process and specify TYPE=EVENT_MAP, you can also generate a stylesheet along with the body file. The stylesheet will tell you which style attributes are being used.

Alternatives to EVENT_MAP

If you want other types of output, here are a few tagsets that you can use as alternatives:

- TEXT_MAP generates more of a listing output.
- TPL_STYLE_LIST (generates HTML) and TPL_STYLE_MAP (generates XML). However, these tagsets list only a subset of the possible attributes.
- STYLE_POPUP generates HTML like HTMLCSS, but if you’re using Internet Explorer, STYLE_POPUP displays a window that shows the resolved ODS style definition for any item that you click.
- STYLE_DISPLAY is like STYLE_POPUP but generates a simple page of output for you to click.
- NAMEDHTML generates HTML output like STYLE_POPUP but with all the objects labeled 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 when creating your own tagsets. Functions can be used on any statement within the tagset language. For information on DATA step functions and statements, see SAS Language Reference: Dictionary and SAS Language Reference: Concepts.

Examples: Creating and Modifying Markup Languages Using the TEMPLATE Procedure

Example 1: Creating a Tagset through Inheritance

PROC TEMPLATE features:
    DEFINE TAGSET statement
        DEFINE EVENT statement
PUT statement
Tagset attribute:
PARENT= attribute

Other ODS features:
ODS PATH statement
ODS MARKUP statement

Program Description
This example defines a new tagset name TAGSET.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 that SAS supplies.

Program

Define a new tagset. The DEFINE TAGSET statement creates a new tagset definition called tagsets.mytags. The PARENT= attribute is used in order for the new tagset tagsets.mytags to inherit events from TAGSETS.CHTML. Note that the ODS PATH statement is specified at the beginning to establish the search path.

```sas
ods path sasuser.templat (update)
    sashelp.tmplmst (read);

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 definition. The system_title event deletes titles.

```sas
define event colspecs;
    put "These are my new colspecs" nl;
end;

define event table;
    put "<p>" nl "<table>";
    finish:
        put "</table>";
end;

define event system_title;
end;
```
End the tagset definition. This END statement ends the tagset definition. The RUN statement executes the PROC TEMPLATE step.

```r
end;
run;
```

Specify the user-defined tagset. The following code tells ODS to use the user-defined tagset TAGSETS.MYTAGS as the tagset definition for the output.

```r
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.

```r
proc print data=sashelp.class;
   run;
```

Stop the creation of the tagset definition. The ODS TAGSET. MYTAGS CLOSE statement closes the MARKUP destination and all the files that are associated with it. You must close the destination before you can view the output with a browser.

```r
ods tagsets.mytags close;
```
To see the customized CHTML tags, view the source from your web browser:

Select from your browser's tool bar:

- View
- Source

These are my new colspecs

<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>

Use the tagset TAGSETS.CHTML that is provided by SAS. To compare the output from TAGSETS.MYTAGS to the TAGSETS.CHTML that is supplied by SAS, the following ODS code specifies the SAS tagset. Note that you can specify any tagset by using TYPE= in an ODS MARKUP statement.

```plaintext
ods markup type=tagsets.chtml body='default-tagset-filename.html';
proc print data=sashelp.class;
run;
ods markup close;
```
Display 11.2  A Display That Uses the Default CHTML Tagset (Viewed with Microsoft Internet Explorer)

To see the default CHTML tags, view the source from your web browser:

☐  Select from your browser's tool bar:

```
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

PROC TEMPLATE features:

SOURCE statement
DEFINE TAGSET
DEFINE EVENT

Program Description

This example copies the source for a tagset which SAS supplies, modifies the definition, then builds a new tagset definition for custom output. To create a new tagset, you can use the SOURCE statement in PROC TEMPLATE to copy a tagset’s source. Then you can customize the definition as needed.

Program

Copy the SAS tagset to an external file. The following statements copy the tagset definition source from the SAS tagset TAGSETS.CSV to the SAS log.

```sas
proc template;
   source tagsets.csv;
run;
```

Output 11.3 CSV Tagset Definition Source

This is the default CSV tagset definition that SAS supplies.

```sas
define tagset Tagsets.Csv;
   notes "This is the CSV definition";
   define event put_value;
      put VALUE;
   end;
   define event put_value_cr;
      put VALUE NL;
   end;
   define event table;
      finish:
         put NL;
   end;
   define event row;
      finish:
         put NL;
   end;
```
define event header;
  start:
    put "," /if `cmp( COLSTART, "1");
    put "";
    put VALUE;
  finish:
    put "";
end;

define event data;
  start:
    put "," /if `cmp( COLSTART, "1");
    put "";
    put VALUE;
  finish:
    put "";
end;

define event colspanfill;
  put ",
end;

define event rowspanfill;
  put "," /if `exists( VALUE);
end;

define event breakline;
  put NL;
end;

define event splitline;
  put NL;
end;

registered_tm = "(r)";
trademark = "(tm)";
copyright = "(c)";
output_type = "csv";
stacked_columns = OFF;
end;

Create your new customized tagset. Submit the following PROC TEMPLATE code to create your new customized tagset tagsets.mycsv. The DEFINE EVENT TABLE statement adds two blank lines to the output file by using the PUT NL statements. One blank line is placed before the table and the other is placed after the table.

define tagset Tagsets.mycsv;
  notes "This is the My CSV definition";
  define event table;
    start:
      put nl;
    finish:
      put nl;
  end;
  define event put_value;
put VALUE;
end;
define event put_value_cr;
    put VALUE NL;
end;
define event row;
    finish:
        put NL;
end;
define event header;
    start:
        put "," /if `cmp( COLSTART, "1");
        put "";
        put VALUE;
        finish:
            put "";
end;
define event data;
    start:
        put "," /if `cmp( COLSTART, "1");
        put "";
        put VALUE;
        finish:
            put "";
end;
define event colspanfill;
    put ",";
end;
define event rowspanfill;
    put "," /if `exists( VALUE);
end;
define event breakline;
    put NL;
end;
define event splitline;
    put NL;
end;
registered_tm = "(r)";
trademark = "(tm)";
copyright = "(c)";
output_type = "csv";
stacked_columns = OFF;
end;
To view the customized CSV Tagsets.mycsv, submit the following code:

```sas
proc template;
  source tagsets.mycsv;
run;
```

```
proc template;
  define tagset Tagsets.Mycsv / store = SASUSER.TEMPLAT;
    notes "This is the My CSV definition";
    define event table;
      start:
        put NL;
        finish:
        put NL;
    end;
    define event put_value;
      put VALUE;
    end;
    define event put_value_cr;
      put VALUE NL;
    end;
    define event row;
      finish:
        put NL;
    end;
    define event header;
      start:
        put "," /if ^cmp( COLSTART, "1");
        put "";
        put VALUE;
      finish:
        put "";
    end;
    define event data;
      start:
        put "," /if ^cmp( COLSTART, "1");
        put "";
        put VALUE;
      finish:
        put "";
    end;
    define event colspanfill;
      put ",";
    end;
    define event rowspanfill;
      put "," /if ^exists( VALUE);
    end;
    define event breakline;
      put NL;
    end;
    define event splitline;
      put NL;
    end;
    output_type = "csv";
    copyright = "(c)";
    trademark = "(tm)";
    registered_tm = "(r)";
    stacked_columns = OFF;
  end;
run;
```
Example 3: Creating a New Tagset

PROC TEMPLATE features:
- DEFINE TAGSET statement
- NOTES statement
- DEFINE EVENT statement
- NDENT statement
- PUT statement
- TRIGGER statement
- XDENT statement

Tagset Attributes:
- DEFAULT_EVENT attribute
- INDENT= attribute
- OUTPUT_TYPE attribute
- MAP= attribute
- MAPSUB= attribute
- NOBREAKSPACE= attribute
- SPLIT= attribute
- STACKED_COLUMNS= attribute

Program Description

This example shows a new tagset definition that does not inherit events from another tagset definition. This is a customized tagset definition for specific PROC FREQ output.

Program

Create the new tagset Tagsets.newloc. The DEFINE TAGSET statement creates a new tagset Tagsets.newloc and specifies where you want to store the tagset.

```sas
proc template;
  define tagset Tagsets.newloc / store = SASUSER.TEMPLAT;
  notes "This is the Location Report Definition";

Define seven events. The seven DEFINE statements create the events named basic, doc, system_title, header, data, country, and frequency.

define event basic;
end;

define event doc;
start:
  put "" nl nl;
  put "" nl;
  put "" nl;
  put "" nl;
  ndent;
finish:
  xdent;
```
put nl;
put ""
end;

define event system_title;
put ""
put VALUE;
put ""
put nl nl;
end;
define event header;
start:
trigger country /if cmp(LABEL, "EmpCountry");
end;

define event data;
start:
trigger frequency /if cmp(name, "Frequency");
end;

define event country;
put "" nl;
indent;
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;
**Example 4: Executing Events Using the TRIGGER= Statement**

PROC TEMPLATE features:

- DEFINE TAGSET statement
- DEFINE EVENT statement
- PUT statement

---

```sas
proc template;
    define tagset Tagsets.Newloc / store = SASUSER.TEMPLAT;
        notes "This is the Location Report Definition";
        define event basic;
        end;
    define event doc;
        start:
            put "" NL NL;
            put "" NL;
            put "" NL;
            put "" NL;
            indent;
        finish:
            xdent;
            put NL;
            put "";
        end;
    define event system_title;
        put "";
        put VALUE;
        put "";
        put NL NL;
    end;
    define event header;
        start:
            trigger country /if cmp( LABEL, "EmpCountry");
        end;
    define event data;
        start:
            trigger frequency /if cmp( name, "Frequency");
        end;
    define event country;
        put "" NL;
        indent;
        put "";
        put VALUE;
        put "" NL;
    end;
    define event frequency;
        put "";
        put VALUE;
        put "" NL;
        xdent;
        put "" NL;
    end;
    map = %nrstr("<>&");
    mapsub = %nrstr("//&/");
    nobreakspace = " ";
    split = " ";
    indent = 2;
    default_event = "basic";
    output_type = "xml";
    stacked_columns = OFF;
end;
run;
```
TRIGGER statement

Other ODS features:
- ODS `directory.tagset-name` statement

Program Description

This example illustrates how to execute events.

Program

Execute different events. The TRIGGER statement executes another event. For example, the start section of DOC triggers the start section of MYTEST and OTHEREVENTA. MYTEST has a start section, so output is generated. OTHEREVENTA 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;
```
Example 5: Indenting Your Output

PROC TEMPLATE features:
  DEFINE TAGSET statement
  DEFINE EVENT statement
    PUT statement
    NDENT statement
    TRIGGER statement
    XDENT statement

TAGSET attributes:
  INDENT= attribute

Other ODS features:
  ODS directory.tagset-name statement

Program Description

This example illustrates how to indent your output using a tagset.

Note: When you view a file with an extension of .xml in an XML-compliant browser, any indention in the file is ignored by the browser in favor of its own indention algorithm. △
Set your beginning indention level and then proceed to increment your indention levels. The INDENT= tagset definition attribute determines how much the NDENT and XDENT event statements indent output.

proc template;
   define tagset tagsets.mytagset2;
      indent = 4;

      define event doc;
         start:
            put "start of doc" nl;
            ndent;
            trigger mytest;
            trigger otherevent;
         finish:
            trigger mytest;
            xdent;
            put "finish of doc" nl;
            trigger mytest start;
            trigger otherevent;
            trigger mytest finish;
      end;

      define event mytest;
         start:
            put "start of mytest" nl;
            ndent;
         finish:
            xdent;
            put "finish of mytest" nl;
      end;

      define event otherevent;
         put "This is my other event" nl;
      end;
   end;
run;
ods tagsets.mytagset2 file='custom-tagset-filename.txt';
ods tagsets.mytagset2 close;
Example 6: Using Different Styles for Events

PROC TEMPLATE features:
- DEFINE EVENT statement
- PUT statement
- TRIGGER statement

Event attribute:
- STYLE= attribute

Program Description
This example shows you how to use different styles for events.

Program

Specify the event definitions. The following event definitions 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>";
   put "<tr>" nl;
finish:
   put "</tr>" nl;
   put "</table>" nl;
```
define event GBanner;
    put "" nl;
    trigger pre_post;
    put "" nl;
end;

define event GNContent;
    put "";
    trigger pre_post start;
    put VALUE;
    trigger pre_post finish;
    put "";
end;

define event noteBanner;
    style="NoteBanner";
    trigger GBanner;
end;

define event NoteContent;
    style="NoteContent";
    trigger GNContent;
end;

define event note;
    trigger Gnote start;
    trigger noteBanner;
    trigger noteContent;
    trigger Gnote finish;
end;

define event WarnBanner;
    style="WarnBanner";
    trigger GBanner;
end;

define event WarnContent;
    style="WarnContent";
    trigger GNContent;
end;

define event Warning;
    trigger Gnote start;
    trigger WarnBanner;
    trigger WarnContent;
    trigger Gnote finish;
end;
Example 7: Modifying an Event to Include Other Stylesheets

PROC TEMPLATE features:
  DEFINE EVENT statement
  PUTQ statement

Program Description

The following program provides some example code that you can use to link previously created stylesheet to an event that you define.

Program

Define an event that links to a stylesheet. This code shows you how to define an event that creates a link to a previously created stylesheet instead of the SAS generated stylesheet.

```sas
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: Creating Different Data Delimiters in a Tagset

PROC TEMPLATE features:
  DEFINE TAGSET statement
  DEFINE EVENT statement
  PUT statement
  NOTES statement
  Tagset attributes:
    OUTPUT_TYPE= attribute
    PARENT= attribute
    STACKED_COLUMNS= attribute

Other ODS features:
  ODS directory.tagset-name statement
  ODS directory.tagset-name CLOSE statement

Data set: GRAIN_PRODUCTION

Program Description

This example creates a customized tagset `tagset.semisv` which inherits attributes from the CSV tagset that SAS provides. This program deletes all the events that do not
have a comma, keeps all the events that do have commas, and then changes all the commas to semicolons.

**Program**

Use the SAS provided tagset definition `tagsets.csv`. `Tagsets.csv` is the tagset that SAS provides to produce tabular output that contains columns of data values, which are separated by commas. The following code is the template that is used to create the tagset `tagsets.csv`.

```sas
proc template;
define tagset Tagsets.Csv;
  notes "This is the CSV definition";
define event put_value;
  put VALUE;
  put NL /if cmp( htmlclass, "batch");
end;
define event table;
  finish:
    put NL;
end;
define event row;
  finish:
    put NL;
end;
define event header;
  start:
    put "," /if "cmp( COLSTART, "1");
    put "=";
    put VALUE;
  finish:
    put "=";
end;
define event data;
  start:
    put "," /if "cmp( COLSTART, "1");
    put "=";
    put VALUE;
  finish:
    put "=";
end;
define event colspanfill;
  put ",";
end;
define event rowspanfill;
  put "," /if "exists( VALUE);";
end;
define event breakline;
  put NL;
end;
define event splitline;
  put NL;
end;
registered_tm = "(r)";
trademark = "(tm)";
copyright = "(c)";
```
output_type = "csv";
stacked_columns = OFF;
end;
run;

Create a new tagset `tagsets.semisv` from the parent tagset `tagsets.csv`. The DEFINE TAGSET statement creates a new tagset `tagsets.semisv`. The new tagset inherits its attributes from the parent tagset `tagsets.csv` which SAS provides. The NOTES statement adds information about the tagset which becomes part of the compiled tagset definition.

```
proc template;
  define tagset tagsets.semisv;
    notes "This is the SEMI-CSV definition";
    parent = tagsets.csv;
  end;
run;
```

Define four events that insert semicolon delimiters. The four DEFINE EVENT statements create the events `header`, `data`, `colspanfill`, `rowspanfill`. The PUT statements insert a semicolon between each column, and enclose each table cell value with quotation marks.

```
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;
end;
run;
```
Specify the user-defined tagset. The following code tells ODS to use the user-defined tagset TAGSETS.SEMISV as the tagset definition for the output.

```ods tagsets.semisv file='custom-tagset-filename.html';
```

Print the data set. PROC PRINT creates the report. ODS writes the report to the body file.

```proc print data=grain_production label;
run;
```

Stop the creation of the tagset definition. The ODS TAGSET. SEMISV CLOSE statement closes the MARKUP destination and all the files that are associated with it. You must close the destination before you can view the output with a viewer.

```ods tagsets.semisv close;
```
Panel 11.5 Grain Production Report Displayed Using a Customized Tagset $\text{tagsets}.\text{semisv}$ (Viewed with Microsoft Excel)
Example 9: Using the STACKED_COLUMNS Attribute in a Tagset Definition

PROC TEMPLATE features:
  DEFINE TABLE statement
  NOTES statement
  COLUMN statement
  DEFINE statement (for columns)

DEFINE TAGSET
  Tagset attribute:
    PARENT= attribute
    STACKED_COLUMNS= attribute

Other ODS features:
  ODS directory.tagset-name statement
  ODS PHTML statement
  ODS _ALL_ CLOSE statement

Program Description

This example shows the difference between stacking data one column on top of another, or placing data side by side. (For more information on stacked columns, see the “DEFINE TABLE Statement” on page 410.)

Program

Create a table definition. The DEFINE TABLE statement creates the table definition Base.Standard.

```sas
proc template;
  define table Base.Standard;
    notes "Table definition for PROC Standard.";
    column name (mean std) n label;
    define name; header="Name" varname="Name" style=RowHeader; end;
    define mean; header="Mean/Std Dev" varname="Mean" format=D12.; end;
    define std; header="/Standard/Deviation" varname="stdDev" format=D12.; end;
    define n; header="N" format=best.; end;
    define label; header="Label" varname="Label"; end;
    byline wrap required_space=3;
  end;
run;
proc template;
  define tagset tagsets.myhtml;
    parent=tagsets.phtml;
    stacked_columns=no;
  end;
run;
```
Customize the tagset by stacking the values side by side. This customized tagset has STACKED_COLUMNS= NO. Note that the SAS tagset, TAGSETS.PHTML, has STACKED_COLUMNS=YES.

```
proc template;
  define tagset tagsets.myhtml;
    parent=tagsets.phtml;
    stacked_columns=no;
  end;
run;
```

Create HTML output and specify the location for storing the HTML output. The ODS TAGSETS.MYHTML statement opens the markup language destination and creates the HTML output. The output objects are sent to the external file `not_stacked.html` in the current directory. The PROC STANDARD statement generates the statistics for the `sashelp.class` data set. The PRINT option prints the report.

```
ods tagsets.myhtml file="not_stacked.html";
proc standard print data=sashelp.class;
run;
```

Stop the creation of the HTML output. The ODS _ALL_ CLOSE statement closes all open destinations and all files associated with them. For HTML output, you must close the HTML destination before you can view the output with a browser.

```
ods _all_ close;
```

Display 11.6 Output with Values Side by Side

### The SAS System

#### The STANDARD Procedure

<table>
<thead>
<tr>
<th>Name</th>
<th>Mean/Std Dev</th>
<th>Standard Deviation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>13.315789</td>
<td>1.492672</td>
<td>19</td>
</tr>
<tr>
<td>Height</td>
<td>62.336842</td>
<td>5.127075</td>
<td>19</td>
</tr>
<tr>
<td>Weight</td>
<td>100.026316</td>
<td>22.773933</td>
<td>19</td>
</tr>
</tbody>
</table>
Create the same file but with values stacked. 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;
```

Display 11.7 Output with Values Stacked One on Top of the 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>
Appendices

Appendix 1. Example Programs 615
Appendix 2. ODS and the HTML Destination 637
Appendix 3. ODS HTML Statements for Running Examples in Different Operating Environments 649
Appendix 4. HTML, Printer Family, and Markup Languages Style Elements and Their Inheritances 651
Appendix 5. Recommended Reading 663
Creating the Employee_data Data Set

options source pagesize=60 linesize=80 nodate;

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

datalines;
  1919 Adams Gerald Stamford CT
  M TA2 34376 15SEP48 07JUN75 203/781-1255
  1653 Alexander Susan Bridgeport CT
  F ME2 35108 18OCT52 12AUG78 203/675-7715
  1400 Apple Troy New York NY
  M ME1 29769 08NOV55 19OCT78 212/586-0808
  1350 Arthur Barbara New York NY
  F FA3 32886 03SEP53 01AUG78 718/383-1549
<table>
<thead>
<tr>
<th>Employee ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Gender</th>
<th>DOB</th>
<th>Hiring Date</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1401</td>
<td>Avery</td>
<td>Jerry</td>
<td>M</td>
<td>16DEC38</td>
<td>20NOV73</td>
<td>201/732-8787</td>
</tr>
<tr>
<td>1499</td>
<td>Barefoot</td>
<td>Joseph</td>
<td>M</td>
<td>29APR42</td>
<td>10JUN68</td>
<td>201/812-5665</td>
</tr>
<tr>
<td>1101</td>
<td>Baucom</td>
<td>Walter</td>
<td>M</td>
<td>09JUN50</td>
<td>04OCT78</td>
<td>212/586-8060</td>
</tr>
<tr>
<td>1333</td>
<td>Blair</td>
<td>Justin</td>
<td>M</td>
<td>02APR49</td>
<td>13FEB69</td>
<td>203/781-1777</td>
</tr>
<tr>
<td>1402</td>
<td>Blalock</td>
<td>Ralph</td>
<td>M</td>
<td>20JAN51</td>
<td>05DEC78</td>
<td>718/384-2849</td>
</tr>
<tr>
<td>1479</td>
<td>Bostic</td>
<td>Marie</td>
<td>M</td>
<td>25DEC56</td>
<td>08OCT77</td>
<td>718/384-8816</td>
</tr>
<tr>
<td>1403</td>
<td>Bowden</td>
<td>Earl</td>
<td>M</td>
<td>28JAN57</td>
<td>24DEC79</td>
<td>203/675-3434</td>
</tr>
<tr>
<td>1399</td>
<td>Boyce</td>
<td>Jonathan</td>
<td>M</td>
<td>28DEC52</td>
<td>30JAN79</td>
<td>212/587-1247</td>
</tr>
<tr>
<td>1658</td>
<td>Bradley</td>
<td>Jeremy</td>
<td>M</td>
<td>11APR55</td>
<td>03MAR80</td>
<td>212/587-3622</td>
</tr>
<tr>
<td>1428</td>
<td>Brady</td>
<td>Christine</td>
<td>F</td>
<td>07APR58</td>
<td>19NOV79</td>
<td>203/781-1212</td>
</tr>
<tr>
<td>1782</td>
<td>Brown</td>
<td>Jason</td>
<td>M</td>
<td>28JAN45</td>
<td>14APR66</td>
<td>212/587-9000</td>
</tr>
<tr>
<td>1244</td>
<td>Bryant</td>
<td>Leonard</td>
<td>M</td>
<td>03SEP51</td>
<td>20JAN76</td>
<td>718/383-3334</td>
</tr>
<tr>
<td>1383</td>
<td>Burnette</td>
<td>Thomas</td>
<td>M</td>
<td>28JAN56</td>
<td>23OCT80</td>
<td>718/384-3569</td>
</tr>
<tr>
<td>1574</td>
<td>Cahill</td>
<td>Marshall</td>
<td>M</td>
<td>28JAN56</td>
<td>23OCT80</td>
<td>718/384-3569</td>
</tr>
<tr>
<td>1789</td>
<td>Caraway</td>
<td>Davis</td>
<td>M</td>
<td>14APR66</td>
<td>212/587-9000</td>
<td></td>
</tr>
<tr>
<td>1404</td>
<td>Carter</td>
<td>Donald</td>
<td>M</td>
<td>03SEP51</td>
<td>20JAN76</td>
<td>718/383-3334</td>
</tr>
<tr>
<td>1437</td>
<td>Carter</td>
<td>Dorothy</td>
<td>F</td>
<td>23SEP48</td>
<td>03SEP72</td>
<td>203/675-4117</td>
</tr>
<tr>
<td>1639</td>
<td>Carter</td>
<td>Karen</td>
<td>F</td>
<td>29JUN48</td>
<td>31JAN72</td>
<td>203/781-8839</td>
</tr>
<tr>
<td>1269</td>
<td>Caston</td>
<td>Franklin</td>
<td>M</td>
<td>28JAN45</td>
<td>01DEC70</td>
<td>203/781-3335</td>
</tr>
<tr>
<td>1065</td>
<td>Chapman</td>
<td>Neil</td>
<td>M</td>
<td>06MAY60</td>
<td>01DEC80</td>
<td>718/384-5618</td>
</tr>
<tr>
<td>1876</td>
<td>Chin</td>
<td>Jack</td>
<td>M</td>
<td>23MAY46</td>
<td>30APR73</td>
<td>212/588-5634</td>
</tr>
<tr>
<td>1037</td>
<td>Chow</td>
<td>Jane</td>
<td>F</td>
<td>13APR52</td>
<td>16SEP80</td>
<td>203/781-8868</td>
</tr>
<tr>
<td>1129</td>
<td>Cook</td>
<td>Brenda</td>
<td>M</td>
<td>11DEC49</td>
<td>20AUG79</td>
<td>718/383-2313</td>
</tr>
<tr>
<td>1988</td>
<td>Cooper</td>
<td>Anthony</td>
<td>M</td>
<td>03DEC47</td>
<td>21SEP72</td>
<td>212/587-1228</td>
</tr>
<tr>
<td>1405</td>
<td>Davidson</td>
<td>Jason</td>
<td>M</td>
<td>29JAN32</td>
<td>10JAN75</td>
<td>718/384-5618</td>
</tr>
<tr>
<td>1430</td>
<td>Dean</td>
<td>Sandra</td>
<td>M</td>
<td>08MAR54</td>
<td>29JAN80</td>
<td>201/732-2323</td>
</tr>
<tr>
<td>1983</td>
<td>Dean</td>
<td>Sharon</td>
<td>F</td>
<td>03MAR50</td>
<td>30APR75</td>
<td>203/675-1647</td>
</tr>
<tr>
<td>1403</td>
<td>FA3</td>
<td>33419</td>
<td>F</td>
<td>03MAR50</td>
<td>30APR75</td>
<td>718/384-1647</td>
</tr>
</tbody>
</table>
Creating the Charity Data Set

```plaintext
proc format;
  value yrFmt . = "All";
  value $schFmt ' ' = "All ";
data Charity;
  input School $ 1-7 Year 9-12 Name $ 14-20 moneyRaised 22-26
    hoursVolunteered 28-29;
  format moneyRaised dollar8.2;
  format hoursVolunteered f3.0;
  format Year yrFmt.;
  format School $schFmt.;
  label School = "Schools";
  label Year = "Years";
datalines;
Monroe 1992 Allison 31.65 19
Monroe 1992 Barry 23.76 16
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Amount</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1992</td>
<td>Candace</td>
<td>21.11</td>
<td>5</td>
</tr>
<tr>
<td>1992</td>
<td>Danny</td>
<td>6.89</td>
<td>23</td>
</tr>
<tr>
<td>1992</td>
<td>Edward</td>
<td>53.76</td>
<td>31</td>
</tr>
<tr>
<td>1992</td>
<td>Fiona</td>
<td>48.55</td>
<td>13</td>
</tr>
<tr>
<td>1992</td>
<td>Gert</td>
<td>24.00</td>
<td>16</td>
</tr>
<tr>
<td>1992</td>
<td>Harold</td>
<td>27.55</td>
<td>17</td>
</tr>
<tr>
<td>1992</td>
<td>Ima</td>
<td>5.98</td>
<td>9</td>
</tr>
<tr>
<td>1992</td>
<td>Jack</td>
<td>20.00</td>
<td>23</td>
</tr>
<tr>
<td>1992</td>
<td>Katie</td>
<td>22.11</td>
<td>2</td>
</tr>
<tr>
<td>1992</td>
<td>Lisa</td>
<td>18.34</td>
<td>17</td>
</tr>
<tr>
<td>1992</td>
<td>Tonya</td>
<td>55.16</td>
<td>40</td>
</tr>
<tr>
<td>1992</td>
<td>Max</td>
<td>26.77</td>
<td>34</td>
</tr>
<tr>
<td>1992</td>
<td>Ned</td>
<td>28.43</td>
<td>22</td>
</tr>
<tr>
<td>1992</td>
<td>Opal</td>
<td>32.66</td>
<td>14</td>
</tr>
<tr>
<td>1993</td>
<td>Patsy</td>
<td>18.33</td>
<td>18</td>
</tr>
<tr>
<td>1993</td>
<td>Quentin</td>
<td>16.89</td>
<td>15</td>
</tr>
<tr>
<td>1993</td>
<td>Randall</td>
<td>12.98</td>
<td>17</td>
</tr>
<tr>
<td>1993</td>
<td>Sam</td>
<td>15.88</td>
<td>5</td>
</tr>
<tr>
<td>1993</td>
<td>Tyra</td>
<td>21.88</td>
<td>23</td>
</tr>
<tr>
<td>1993</td>
<td>Myrtle</td>
<td>47.33</td>
<td>26</td>
</tr>
<tr>
<td>1993</td>
<td>Frank</td>
<td>41.11</td>
<td>22</td>
</tr>
<tr>
<td>1993</td>
<td>Cameron</td>
<td>65.44</td>
<td>14</td>
</tr>
<tr>
<td>1993</td>
<td>Vern</td>
<td>17.89</td>
<td>11</td>
</tr>
<tr>
<td>1993</td>
<td>Wendell</td>
<td>23.00</td>
<td>10</td>
</tr>
<tr>
<td>1993</td>
<td>Bob</td>
<td>26.88</td>
<td>6</td>
</tr>
<tr>
<td>1993</td>
<td>Leah</td>
<td>28.99</td>
<td>23</td>
</tr>
<tr>
<td>1994</td>
<td>Becky</td>
<td>30.33</td>
<td>26</td>
</tr>
<tr>
<td>1994</td>
<td>Sally</td>
<td>35.75</td>
<td>27</td>
</tr>
<tr>
<td>1994</td>
<td>Edgar</td>
<td>27.11</td>
<td>12</td>
</tr>
<tr>
<td>1994</td>
<td>Dawson</td>
<td>17.24</td>
<td>16</td>
</tr>
<tr>
<td>1994</td>
<td>Lou</td>
<td>5.12</td>
<td>16</td>
</tr>
<tr>
<td>1994</td>
<td>Damien</td>
<td>18.74</td>
<td>17</td>
</tr>
<tr>
<td>1994</td>
<td>Mona</td>
<td>27.43</td>
<td>7</td>
</tr>
<tr>
<td>1994</td>
<td>Della</td>
<td>56.78</td>
<td>15</td>
</tr>
<tr>
<td>1994</td>
<td>Monique</td>
<td>29.88</td>
<td>19</td>
</tr>
<tr>
<td>1994</td>
<td>Carl</td>
<td>31.12</td>
<td>25</td>
</tr>
<tr>
<td>1994</td>
<td>Reba</td>
<td>35.16</td>
<td>22</td>
</tr>
<tr>
<td>1994</td>
<td>Dax</td>
<td>27.65</td>
<td>23</td>
</tr>
<tr>
<td>1994</td>
<td>Gary</td>
<td>23.11</td>
<td>15</td>
</tr>
<tr>
<td>1994</td>
<td>Suzie</td>
<td>26.65</td>
<td>11</td>
</tr>
<tr>
<td>1994</td>
<td>Benito</td>
<td>47.44</td>
<td>18</td>
</tr>
<tr>
<td>1994</td>
<td>Thomas</td>
<td>21.99</td>
<td>23</td>
</tr>
<tr>
<td>1994</td>
<td>Annie</td>
<td>24.99</td>
<td>27</td>
</tr>
<tr>
<td>1994</td>
<td>Paul</td>
<td>27.98</td>
<td>22</td>
</tr>
<tr>
<td>1994</td>
<td>Alex</td>
<td>24.00</td>
<td>16</td>
</tr>
<tr>
<td>1994</td>
<td>Lauren</td>
<td>15.00</td>
<td>17</td>
</tr>
<tr>
<td>1994</td>
<td>Julia</td>
<td>12.98</td>
<td>15</td>
</tr>
<tr>
<td>1994</td>
<td>Keith</td>
<td>11.89</td>
<td>19</td>
</tr>
<tr>
<td>1994</td>
<td>Jackie</td>
<td>26.88</td>
<td>22</td>
</tr>
<tr>
<td>1994</td>
<td>Pablo</td>
<td>13.98</td>
<td>28</td>
</tr>
<tr>
<td>1994</td>
<td>L.T.</td>
<td>56.87</td>
<td>33</td>
</tr>
<tr>
<td>1994</td>
<td>Willard</td>
<td>78.65</td>
<td>24</td>
</tr>
<tr>
<td>1994</td>
<td>Kathy</td>
<td>32.88</td>
<td>11</td>
</tr>
<tr>
<td>1994</td>
<td>Abby</td>
<td>35.88</td>
<td>10</td>
</tr>
</tbody>
</table>
Kennedy 1992 Arturo 34.98 14
Kennedy 1992 Grace 27.55 25
Kennedy 1992 Winston 23.88 22
Kennedy 1992 Vince 12.88 21
Kennedy 1992 Claude 15.62 5
Kennedy 1992 Mary 28.99 34
Kennedy 1992 Abner 25.89 22
Kennedy 1992 Jay 35.89 35
Kennedy 1992 Alicia 28.77 26
Kennedy 1992 Freddy 29.00 27
Kennedy 1992 Eloise 31.67 25
Kennedy 1992 Jenny 43.89 22
Kennedy 1992 Thelma 52.63 21
Kennedy 1992 Tina 19.67 21
Kennedy 1992 Eric 24.89 12
Kennedy 1993 Bubba 37.88 12
Kennedy 1993 G.L. 25.89 21
Kennedy 1993 Bert 28.89 21
Kennedy 1993 Clay 26.44 21
Kennedy 1993 Leeann 27.17 17
Kennedy 1993 Georgia 38.90 11
Kennedy 1993 Bill 42.23 25
Kennedy 1993 Holly 18.67 27
Kennedy 1993 Benny 19.09 25
Kennedy 1993 Cammie 28.77 28
Kennedy 1993 Amy 27.08 31
Kennedy 1993 Doris 22.22 24
Kennedy 1993 Robbie 19.80 24
Kennedy 1993 Ted 27.07 25
Kennedy 1993 Sarah 24.44 12
Kennedy 1993 Megan 28.89 11
Kennedy 1993 Jeff 31.11 12
Kennedy 1993 Taz 30.55 11
Kennedy 1993 George 27.56 11
Kennedy 1993 Heather 38.67 15
Kennedy 1994 Nancy 29.90 26
Kennedy 1994 Rusty 30.55 28
Kennedy 1994 Mimi 37.67 22
Kennedy 1994 J.C. 23.33 27
Kennedy 1994 Clark 27.90 25
Kennedy 1994 Rudy 27.78 23
Kennedy 1994 Samuel 34.44 18
Kennedy 1994 Forrest 28.89 26
Kennedy 1994 Luther 72.22 24
Kennedy 1994 Trey 6.78 18
Kennedy 1994 Albert 23.33 19
Kennedy 1994 Che-Min 26.66 33
Kennedy 1994 Preston 32.22 23
Kennedy 1994 Larry 40.00 26
Kennedy 1994 Anton 35.99 28
Kennedy 1994 Sid 27.45 25
Kennedy 1994 Will 28.88 21
Kennedy 1994 Morty 34.44 25
Creating the Statepop Data Set

data statepop;
  input State $ CityPop_80 CityPop_90 NonCityPop_80 NonCityPop_90 Region;
  format region 1.;
  label citypop_80= '1980 metropolitan pop in millions'
  noncitypop_80= '1980 nonmetropolitan pop in millions'
  citypop_90= '1990 metropolitan pop in millions'
  noncitypop_90= '1990 nonmetropolitan pop in million'
  region='Geographic region';
datalines;
ME .405 .443 .721 .785 1
NH .535 .659 .386 .450 1
VT .133 .152 .378 .411 1
MA 5.530 5.788 .207 .229 1
RI .886 .938 .061 .065 1
CT 2.982 3.148 .126 .140 1
NY 16.144 16.515 1.414 1.475 1
PA 10.067 10.083 1.798 1.799 1
DE .496 .553 .098 .113 2
MD 3.920 4.439 .297 .343 2
DC .638 .607 . . . 2
VA 3.966 4.773 1.381 1.414 2
WV .796 .748 1.155 1.045 2
NC 3.749 4.376 2.131 2.253 2
SC 2.114 2.423 1.006 1.064 2
GA 3.507 4.352 1.956 2.127 2
FL 9.039 12.023 .708 .915 2
KY 1.735 1.780 1.925 1.906 2
TN 3.045 3.298 1.546 1.579 2
AL 2.560 2.710 1.334 1.331 2
MS .716 .776 1.805 1.798 2
AR .963 1.040 1.323 1.311 2
LA 3.125 3.160 1.082 1.060 2
OK 1.724 1.870 1.301 1.276 2
TX 11.539 14.166 2.686 2.821 2
OH 8.791 8.826 2.007 2.021 3
IN 3.885 3.962 1.605 1.582 3
IL 9.461 9.574 1.967 1.857 3
MI 7.719 7.698 1.543 1.598 3
WI 3.176 3.331 1.530 1.561 3
MN 2.674 3.011 1.402 1.364 3
IA 1.198 1.200 1.716 1.577 3
MO 3.314 3.491 1.603 1.626 3
ND .234 .257 .418 .381 3
SD .194 .221 .497 .475 3
NE .728 .787 .842 .791 3
KS 1.184 1.333 1.180 1.145 3
MT .189 .191 .598 .608 4
ID .257 .296 .687 .711 4
Creating the Model Data Set

```plaintext
data one;
    input year import doprod stock consum;
    datalines;
49 15.9 149.3 4.2 108.1
50 16.4 161.2 4.1 114.8
51 19.0 171.5 3.1 123.2
52 19.1 175.5 3.1 126.9
53 18.8 180.8 1.1 132.1
54 20.4 190.7 2.2 137.7
55 22.7 202.1 2.1 146.0
56 26.5 212.4 5.6 154.1
57 28.1 226.1 5.0 162.3
58 27.6 231.9 5.1 164.3
59 26.3 239.0 0.7 167.6
60 31.1 258.0 5.6 176.8
61 33.3 269.8 3.9 186.6
62 37.0 288.4 3.1 199.7
63 43.3 304.5 4.6 213.9
64 49.0 323.4 7.0 223.8
65 50.3 336.8 1.2 232.0
66 56.6 353.9 4.5 242.9
;

data model;
    input year 1-2 a 3-9 .3 b 10-17 .3 r4 18-24 .3 r8 25-31 .3
c 32-38 .3 d 39-45 .3 e 46-51 .3 r23 52-58 .3
    r24 59-64 .3 r29 65-70 .3 r33 71-77 .3;
    datalines;
60 994534 53552371656049 9362944261250 8921423631971140299106045 8780 335066
611253576 5580643177015110671424650930 9933453874651217360151507 36871 49192
621318885 621448018932921075688469573610686654502881317293178014 66671 566079
631507969 666125121046261533088511701311673695162821579148179797106485 -4568
641811051 731945021737841454106554095914677245822921945534206255145948 -10940
652532026 816707123363201962785640926221155676314091906268218759195733-145568
```

Programs that Illustrate Inheritance

The programs in this section show the PROC TEMPLATE steps that were used in “About Style Definition Inheritance and Style Element Inheritance” on page 322 to illustrate inheritance in style definitions. These programs also show the SAS code that uses the style definitions.

SAS Program for a Style with One Style Element

This program generates the HTML output in Display 9.3 on page 324.

```sas
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
  input country $ 1-13 grain $ 15-18 kilotons;
  datalines;
  Brazil Rice 10035
  China Rice 190100
  India Rice 120012
  Indonesia Rice 51165
  United States Rice 7771;
proc template;
  define table mytable;
    column x y z;
```
Example Programs

SAS Program for a Style with Two Style Elements (Independently Defined)

This program generates the HTML output in Creating a Second Style Element Independently or from an Existing Style Element on page 324. In this version of the code, the style element `celldataemphasis` is created independently of `celldatasimple`.

```sas
define x;
    style=celldatasimple;
    dataname=country;
    header='Country';
end;
define y;
    style=celldatasimple;
    dataname=grain;
    header='Grain';
end;
define z;
    style=celldatasimple;
    dataname=kilotons;
    header='Kilotons';
end;
end;
run;

proc template;
    /* to ensure a fresh start with the styles */
    delete concepts.style1;
    delete concepts.style2;
run;

proc template;
    define style concepts.style1;
    style celldatasimple /
        font_face=arial
        background=very light vivid blue
        foreground=white;
end;
run;

ods html3 body='display1-body.htm'
    style=concepts.style1;
data _null_; 
    set test;
    file print ods=(template='mytable');
    put _ods_; 
run;
ods html3 close;
```
```sas
input country $ 1-13 grain $ 15-18 kilotons;
datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771
;
proc template;
define table mytable;
column x y z;
define x;
   style=celldatasimple;
dataname=country;
   header='Country';
end;
define y;
   style=celldataemphasis;
dataname=grain;
   header='Grain';
end;
define z;
   style=celldatasimple;
dataname=kilotons;
   header='Kilotons';
end;
end;
run;

proc template;
   /* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;
proc template;
define style concepts.style1;
   style celldatasimple /
      font_face=arial
      background=very light vivid blue
      foreground=white;
   style celldataemphasis from celldatasimple /
      foreground=blue
      font_style=italic;
end;
run;

ods html3 body='display2-body.htm'
   style=concepts.style1;
data _null_
   set test;
   file print ods=(template='mytable');
   put _ods_;
run;
```
**SAS Program for a Style with Two Style Elements (Defined with Inheritance)**

This program generates the HTML output in Creating a Second Style Element Independently or from an Existing Style Element on page 324. In this version of the code, the style element `celldataemphasis` is created as a child of `celldatasimple`.

```sas
ods html3 close;
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771;
proc template;
define table mytable;
column x y z;
define x;
   style=celldatasimple;
dataname=country;
   header='Country';
end;
define y;
   style=celldataemphasis;
dataname=grain;
   header='Grain';
end;
define z;
   style=celldatasimple;
dataname=kilotons;
   header='Kilotons';
end;
end;
run;
proc template;
   /* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;
proc template;
   define style concepts.style1;
      style celldatasimple /
         font_face=arial
         background=very light vivid blue
         foreground=white;
```
SAS Program for Changing the Font Face in Only One Style Element

This program generates the HTML output in Display 9.5 on page 327.

```sas
ods path sashelp.tmplmst(read) sasuser.templat(update);
ods html3 body='display2-body.htm'
   style=concepts.style1;
ods html3 close;

data _null_
   set test;
   file print ods=(template='mytable');
   put _ods_;
run;
ods html3 close;
```

```sas
proc template;
   define table mytable;
      column x y z;
      define x;
         style=celldatasimple;
         dataname=country;
         header='Country';
      end;
      define y;
         style=celldataemphasis;
         dataname=grain;
         header='Grain';
      end;
      define z;
         style=celldatasimple;
         dataname=kilotons;
         header='Kilotons';
      end;
   end;
run;
```

```sas
ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771
;
```

```sas
proc template;
   define table mytable;
      column x y z;
      define x;
         style=celldatasimple;
         dataname=country;
         header='Country';
      end;
      define y;
         style=celldataemphasis;
         dataname=grain;
         header='Grain';
      end;
      define z;
         style=celldatasimple;
         dataname=kilotons;
         header='Kilotons';
      end;
   end;
run;
```

```sas
ods html3 body='display2-body.htm'
   style=concepts.style1;
```

```sas
data _null_
   set test;
   file print ods=(template='mytable');
   put _ods_;
run;
```

```sas
ods html3 close;
```
Example Programs  

/* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;
proc template;
define style concepts.style1;
style celldatasimple
    ‘The change to FONT_FACE does not affect celldataemphasis.’
    / font_face=times
    background=very light vivid blue
    foreground=white;
style celldataemphasis /
    font_face=arial
    background=very light vivid blue
    foreground=blue
    font_style=italic;
end;
run;

ods html3 body='display3-body.htm'
    style=concepts.style1;
data _null_; set test;
    file print ods=(template='mytable');
    put _ods_; run;
oda html3 close;

---

SAS Program for Inheriting a Change to a Style Element

This program generates the HTML output in Display 9.6 on page 327. In this version of the code, the style element **celldataemphasis** is created as a child of **celldatasimple**.

ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
    input country $ 1-13 grain $ 15-18 kilotons;
    datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771
;
proc template;
define table mytable;
    column x y z;
    define x;
        style=celldatasimple;
    dataname=country;

---

SAS Program for Inheriting a Change to a Style Element

This program generates the HTML output in Display 9.6 on page 327. In this version of the code, the style element **celldataemphasis** is created as a child of **celldatasimple**.
SAS Program for Creating the Style Element `celldatalarge`

This program generates the HTML output in Adding the Style Element `celldatalarge` on page 328.

```sas
ods path sashelp.tmplmst(read) sasuser.templat(update);

title;

options nodate pageno=1 linesize=72 pagesize=60;

data test;

input country $ 1-13 grain $ 15-18 kilotons;

run;
```
SAS Program for Creating the Style Element *celldatalarge*

data datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771;

proc template;
define table mytable;
column x y z;
define x;
style=celldatasimple;
dataname=country;
header='Country';
end;
define y;
style=celldataemphasis;
dataname=grain;
header='Grain';
end;
define z;
style=celldatalarge;
dataname=kilotons;
header='Kilotons';
end;
end;
rImplemented with a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
rBuilt styles:
define style concepts.style1;
  style celldatasimple /
    font_face=arial
    background=very light vivid blue
    foreground=white;
  style celldataemphasis from celldatasimple /
    foreground=blue
    font_style=italic;
  style celldatalarge from celldataemphasis /
    font_weight=bold
    font size=5;
end;
rBuilt the HTML template:
ods html body='display5-body.htm'
  style=concepts.style1;
data _null_
  set test;
  file print ods=(template='mytable');
SAS Program for Creating a New Style Element from a Style Element in the Parent Style Definition

This program generates the HTML output in Creating a New Style Element from a Style Element in the Parent Style Definition on page 333.

ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
    input country $ 1-13 grain $ 15-18 kilotons;
datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771
;
proc template;
define table mytable;
column x y z w;
define x;
    style=celldatasimple;
dataname=country;
    header='Country';
end;
define y;
    style=celldataemphasis;
dataname=grain;
    header='Grain';
end;
define z;
    style=celldatalarge;
dataname=kilotons;
    header='Kilotons';
end;
define w;
    style=celldatasmall;
dataname=kilotons;
    header='Kilotons';
end;
end;
run;
proc template;
    /* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;
proc template;
  define style concepts.style1;
    style celldatasimple /
      font_face=arial
      background=very light vivid blue
      foreground=white;
    style celldataemphasis from celldatasimple /
      foreground=blue
      font_style=italic;
    style celldatalarge from celldataemphasis /
      font_weight=bold
      font_size=5;
  end;
run;

proc template;
  define style concepts.style2;
    parent=concepts.style1;
    style celldatasmall from celldatalarge /
      font_size=2;
  end;
run;
ods html3 body='display6-body.htm'
  style=concepts.style2;
data _null_
  set test;
  file print ods=(template='mytable');
  put _ods_;
run;
ods html3 close;

SAS Program for Inheriting Changes to the Parent Style Definition

This program generates the HTML output in Display 9.9 on page 335.

ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
  input country $ 1-13 grain $ 15-18 kilotons;
  datalines;
Brazil Rice  10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771
;
proc template;
  define table mytable;
    column x y z w;
    define x;
      style=celldatasimple;
data name=country;
   header='Country';
end;
define y;
   style=celldataemphasis;
   dataname=grain;
   header='Grain';
end;
define z;
   style=celldatalarge;
   dataname=kilotons;
   header='Kilotons';
end;
define w;
   style=celldatasmall;
   dataname=kilotons;
   header='Kilotons';
end;
end;
run;

proc template;
  /* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;

proc template;
  define style concepts.style1;
      style celldatasimple /
          font_face=times
          background=very light vivid blue
          foreground=white;
      style celldataemphasis from celldatasimple /
          foreground=black
          font_style=italic;
      style celldatalarge from celldataemphasis /
          font_weight=bold
          font_size=5;
end;
run;

proc template;
  define style concepts.style2;
      parent=concepts.style1;
      style celldatasmall from celldatalarge /
          font_size=2;
end;
run;
ods html3 body='display7-body.htm'
    style=concepts.style2;
data _null_
  set test;
  file print ods=(template='mytable');
Example Programs

SAS Program for Using the STYLE Statement to Alter an Existing Style Element in the Child Definition

This program generates the HTML output in Display 9.10 on page 337.

ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
   input country $ 1-13 grain $ 15-18 kilotons;
   datalines;
Brazil       Rice 10035
China        Rice 190100
India        Rice 120012
Indonesia    Rice 51165
United States Rice 7771
;
proc template;
define table mytable;
column x y z w;
define x;
   style=celldatasimple;
dataname=country;
   header='Country';
end;
define y;
   style=celldataemphasis;
dataname=grain;
   header='Grain';
end;
define z;
   style=celldatalarge;
dataname=kilotons;
   header='Kilotons';
end;
define w;
   style=celldatasmall;
dataname=kilotons;
   header='Kilotons';
end;
end;
run;

proc template;
   /* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;
SAS Program for Using the REPLACE Statement to Alter a Style Element and Its Children

This program generates the HTML output in Display 9.11 on page 341.

ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
  input country $ 1-13 grain $ 15-18 kilotons;
datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771;
proc template;
  define table mytable;
    column x y z w;
run;
}

proc template;
  parent=concepts.style1;
  style celldatalarge from celldataemphasis /
    font_weight=bold
    font_size=5;
end;
run;
ods html3 body='display8-body.htm'
  style=concepts.style2;
data _null_
  set test;
  file print ods=(template='mytable');
  put _ods_;
run;
ods html3 close;

SAS Program for Using the REPLACE Statement to Alter a Style Element and Its Children

This program generates the HTML output in Display 9.11 on page 341.

ods path sashelp.tmplmst(read) sasuser.templat(update);
title;
options nodate pageno=1 linesize=72 pagesize=60;
data test;
  input country $ 1-13 grain $ 15-18 kilotons;
datalines;
Brazil Rice 10035
China Rice 190100
India Rice 120012
Indonesia Rice 51165
United States Rice 7771;
proc template;
  define table mytable;
    column x y z w;
run;
ods html3 body='display8-body.htm'
  style=concepts.style2;
data _null_
  set test;
  file print ods=(template='mytable');
  put _ods_;
run;
ods html3 close;
define x;
  style=celldatasimple;
  dataname=country;
  header='Country';
end;
define y;
  style=celldataemphasis;
  dataname=grain;
  header='Grain';
end;
define z;
  style=celldatalarge;
  dataname=kilotons;
  header='Kilotons';
end;
define w;
  style=celldatasmall;
  dataname=kilotons;
  header='Kilotons';
end;
end;
run;

proc template;
  /* to ensure a fresh start with the styles */
delete concepts.style1;
delete concepts.style2;
run;

proc template;
  define style concepts.style1;
    style celldatasimple /
      font_face=arial
      background=very light vivid blue
      foreground=white;
    style celldataemphasis from celldatasimple /
      foreground=blue
      font_style=italic;
    style celldatalarge from celldataemphasis /
      font_weight=bold
      font_size=5;
  end;
run;

proc template;
  define style concepts.style2;
    parent=concepts.style1;
    replace celldataemphasis from celldatasimple /
      foreground=blue
      font_style=italic
      background=white;
    style celldatasmall from celldatalarge /
      font_size=2;
  end;
run;
ods html3 body='display9-body.htm'
    style=concepts.style2;
data _null_
    set test;
    file print ods=(template='mytable');
    put _ods_
run;
ods html3 close;
HTML Links and References Produced by the HTML Destination

What Are Links and References?

An HTML link is a place in a document that allows you to jump to another specific place in the same document or in another document. A browser typically highlights the text that is between the tags that begin and end the link. When you click on the highlighted text, the browser displays the text at the link target. The browser might then display the contents of the target in the active window, or it might open another browser window that displays the contents of the target.

An HTML reference names a file for the browser to display. When a browser reads a reference, it displays the referenced file as if it were part of the file that it is displaying. You can't tell by looking at the browser's display that some of the material is in the file that you are actually viewing and that some is referenced.

When you use ODS, the software automatically creates the links and references that you need. You can, however, customize these links to some extent. If you wish to do so, then you will need to understand how HTML implements links and references.

Implementing HTML Links and References

Note: This simplified discussion of HTML links and references is designed to provide information that will help you understand what ODS does when it builds links and references for you. For a complete discussion of HTML tagging, consult one of the many reference books that are available on the subject. △

Each link in HTML is implemented with a combination of two sets of <A> (anchor) tags. One anchor tag, which is the starting point of the link, has an HREF attribute...
that identifies the anchor tag to link to. The other anchor tag, which is the target of the link, has a NAME attribute. This NAME attribute is what the HREF attribute in the first anchor tag points to. The value of each NAME attribute in a file must be unique so that each value of HREF points to a single, unambiguous location. The following figure illustrates linking within a file:

Figure A2.1  Linking within a File

This <A href="#target1">link</A> points to the anchor tag in this file with NAME="target1".

more text . . .

The target is elsewhere in the file. In fact, it is located <A name="target1">right here</A> in this paragraph.

more text . . .

The important features at the starting point of this link are

- The <A> and </A> tags surround the text that the browser will highlight.
- The HREF attribute points to the link’s target. The target is an anchor tag whose NAME attribute matches the text that follows the pound sign in the HREF attribute. Because no text precedes the pound sign (#), the browser knows that the target is in the same file as the anchor.

When a link points to a target outside the file that is being displayed, the HREF attribute must include the path to that file. The path can be the path within the file system or the uniform resource locator (URL) of the file. The following figure illustrates a link from one file to another file that is specified with a URL:
The browser highlights the word **link**. When you click on **link**, the browser positions the target **right here** in the active window or opens another window that displays the target.

File: /users/brown/documents/file1

URL: http://www.company-url/local-url/file1

This

`<A href="http://www.company-url/local-url/file2#target1">link</A>`

points to an anchor tag in the file with the specified URL. The NAME attribute on the target anchor tag is "target1".

File: /users/brown/documents/file2

URL: http://www.company-url/local-url/file2

The target is in this file. In fact, it is located

`<A name="target1">right here</A>`

in this sentence.

The important features at the starting point (the anchor) of the link are

- The `<A>` and `</A>` tags surround the text that the browser will highlight.
- The HREF attribute points to the link’s target. The text that precedes the pound sign (#) identifies the file that contains the target.

ODS provides features that enable you to customize the text that precedes the pound sign and the text that follows the pound sign. For information on how to do this, see the discussions of `file-specification`, `ANCHOR=`, `BASE=`, `PATH=`, and `GPATH=` in the “ODS HTML Statement” on page 95 as well as “How ODS Constructs Links and References” on page 640.

HTML implements references in much the same way as it implements links. The main difference is that a link points to a particular location within a file and that a reference points to the file itself. HTML uses the SRC attribute to identify a file to reference. The value of the SRC attribute is constructed the same way that the value of the HREF attribute is constructed except that there is no pound sign and no text following it.
How ODS Constructs Links and References

Several options in the ODS HTML statement affect how ODS constructs the links and references that point from the frame to the table of contents, table of pages, and body file and from the table of contents or table of pages to the body file. Links are made as HREF attributes on <A> (anchor) tags inside the HTML files. Each HREF attribute points to the NAME attribute on another <A> tag. The HREF must identify both the file that contains the target and the name of the anchor within that file. The value of HREF must be a valid target in a valid URL. It uses the following form:

```
<A href="URL#anchor-name">
```

ODS constructs the value of an HREF attribute based on information that you provide in the ODS HTML statement.

Note: HTML references to files use other tags, but the logic for creating the string that identifies the file is the same as the logic for creating an HREF attribute (see “How ODS Constructs Links and References” on page 640).

The URL in an HREF attribute is composed of information from three options in the ODS HTML statement: the BASE option; the GPATH= or the PATH= option; and the BODY=, the CONTENTS=, or the PAGE= option.

1. If you specify BASE=, then the value of that option is the first part of the URL for every HREF attribute that ODS writes.

2. If you specify GPATH= or PATH=, then the next part of the URL in an HREF attribute comes from that option.

If the file that you are linking to is a high-resolution graphic, then ODS uses information from the GPATH= option as the next part of the HREF. For information on these options, see the discussion of GPATH= and the discussion of PATH= in the “ODS HTML Statement” on page 95. The following table shows how ODS uses information from the GPATH= option in the URL in HREF attributes:

<table>
<thead>
<tr>
<th>If the file-specification in GPATH= is ...</th>
<th>And the URL= suboption is ...</th>
<th>then ODS uses this information in the second part of the URL in the HREF attribute ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>an external-file or libref:catalog</td>
<td>not specified</td>
<td>the name of the file</td>
</tr>
<tr>
<td>an external-file or libref:catalog</td>
<td>specified, but not NONE</td>
<td>the value of the URL= suboption</td>
</tr>
<tr>
<td>an external-file or libref:catalog</td>
<td>NONE</td>
<td>No information from GPATH=</td>
</tr>
<tr>
<td>a fileref</td>
<td>specified or not specified</td>
<td>No information from GPATH=</td>
</tr>
</tbody>
</table>

* If you do not specify GPATH=, then ODS uses the value of PATH= to create this part of the HREF.

If the file that you are linking to is not a high-resolution graphic, then ODS uses information from the PATH= option as the next part of the HREF. The following table shows how ODS uses information from the PATH= option in the URL in HREF attributes:
Table A2.2  Building an HREF Attribute from the PATH= Option

<table>
<thead>
<tr>
<th>file-specification</th>
<th>URL=suboption</th>
<th>Information used in the second part of the URL in the HREF attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>external-file or libref.catalog</td>
<td>not specified</td>
<td>the name of the file</td>
</tr>
<tr>
<td>external-file or libref.catalog</td>
<td>specified, but not NONE</td>
<td>the value of the URL= suboption</td>
</tr>
<tr>
<td>external-file or libref.catalog</td>
<td>NONE</td>
<td>No information from PATH=</td>
</tr>
<tr>
<td>fileref</td>
<td>specified or not specified</td>
<td>No information from PATH=</td>
</tr>
</tbody>
</table>

Note: If you use a fileref as the file specification in the BODY=, CONTENTS=, or PAGE= option in the ODS HTML statement, and you do not use the URL= suboption in that option, then ODS does not use information from GPATH= or PATH= when it creates the complete URL for any corresponding HREF attributes.

3 The last part of the URL that is used in an HREF attribute is, by default, the name of the file that contains the target. ODS determines the name of the file from the file-specification that you use in the BODY=, CONTENTS=, or PAGE= option. (ODS does not create links or references to frame files.) For more information on these options, see the discussion of file-specification on page 113.)

If you specify the URL= suboption in one of these options, then ODS uses the string that you specify instead of the file name.

Note: If you use a fileref as the file specification and do not use the URL= suboption, then ODS does not use information from GPATH= or PATH= when it creates the complete URL for the HREF attribute.

The anchor-name comes from the value of the ANCHOR= option.

The following figure illustrates the creation of the HREF:
Files Produced by the HTML Destination

The HTML destination can produce four kinds of files: body, contents, frame, and page files. You create these files with options in the ODS HTML statement (see “ODS HTML Statement” on page 95 for details).

The Body File

The body file contains HTML output that is generated from the output objects that your SAS job creates. The style definition and the table definition that the job uses determine the appearance and content of the tables and the cells within them.

Typically, when you route an output object that does not contain graphics to the HTML destination, ODS places the results within TABLE tags, generating them as one or more HTML tables.

Graphics output is produced according to the SAS code that generates it. Instead of using <TABLE> tags, the body file contains an <IMG> (image) tag that references the graphic. When you view the body file in a browser, you cannot tell that the graphic is not part of the body file because the <IMG> tag displays it in the browser.

Note: A very few procedures produce output objects that are neither tabular nor graphics. In these cases, the output is not tagged as an HTML table. △

Titles and footnotes in the body file are generated as HTML tables of their own near the top and bottom of each page of HTML output.

Note: For graphics output, titles and footnotes are, by default, part of the graphics file. You can use the NOGTITLE and NOGFOOTNOTE options to place them in the body file instead. See the discussion of GTITLE and GFOOTNOTE in “ODS HTML Statement” on page 95 for more information. △

All <TABLE> tags and all <IMG> tags are potential targets for links or references (see “How ODS Constructs Links and References” on page 640). Therefore, ODS must provide an <A> tag with a NAME attribute close to each <TABLE> and <IMG> tag for links and references to point to. The NAME attribute on the anchor tag becomes the final part of any reference or link to the table. ODS inserts anchor tags in its HTML output as follows:

- ODS places an anchor tag near the top of each page, before all tables on the page (including the table that holds the titles) and before all images. This anchor is the target for links to the first table (excluding any titles) or to the first image on the page.

  Note: Each procedure or DATA step starts a new page. In addition, ODS produces a new page of output whenever the SAS program explicitly asks for a new page. For example, if you use the page dimension in PROC TABULATE, then you create a page for each value of the variable that defines the pages. In this context, the word page has nothing to do with the PAGESIZE= setting in your SAS session. △

- ODS places an anchor tag slightly before each <TABLE> tag, provided that the table contains results (not titles or footnotes) and that it is not the first table or image on the page.

- ODS places an anchor tag slightly before each <IMG> tag, provided that it is not the first table or image on a page.
The following figure illustrates the placement of anchor tags from a SAS job that executes two procedures. The first procedure creates two HTML tables of results on a single page. The page also includes an HTML table for the title and one for the footnote. Solid arrows indicate which `<A>` tag ODS uses as a target for each table. The second procedure creates a GIF file. The titles for this procedure are part of the GIF file (the default behavior). Again, the solid arrow indicates which anchor tag ODS uses as a target when it creates a link to the image. The dashed arrow points to the file that the `<IMG>` tag references.
Figure A2.4   Placement of <A> (anchor) Tags in HTML Output

First Page of Body File

<A name="IDX"></A>

table for titles

<TABLE...>
first table of results
</TABLE>

<A name="IDX1"></A>

<TABLE...>
second table of results
</TABLE>

table for footnotes

Second Page of Body File

<A name="IDX2"></A>

<IMG src="gmap1.gif">
For a view of this same file through a browser, see Browser View of HTML Frame File Display A2.1 on page 647.

---

**The Contents File**

The contents file contains a link to the body file for each HTML table that ODS creates from procedure or DATA step results. The targets for these links are the values of the NAME attributes on the anchor tags that are in the body file, see “The Body File” on page 642. For example, an anchor tag that links to the second HTML table of results in Placement of `<A>` (anchor) Tags in HTML Output on page 644 looks like this:

```html
<A href="pop-body.htm#IDX1">
```

In this anchor tag

- `pop-body.htm` identifies the file that contains the target.
- `#IDX1` provides the name of the target.

You can view the contents file directly in the browser, or, if you make a frame file, you can see the contents file as part of the frame file (see “The Frame File” on page 645).

---

**The Page File**

The page file contains a link to the body file for each page of HTML output that ODS creates from procedure or DATA step results. The targets for these links are the values of the NAME attributes on the anchor tags that are in the body file (see “The Body File” on page 642). For example, an anchor tag that links to the second page of results in Placement of `<A>` (anchor) Tags in HTML Output on page 644 looks like this:

```html
<A href="pop-body.htm#IDX2">
```

In this anchor tag

- `pop-body.htm` identifies the file that contains the target.
- `#IDX2` provides the name of the target.

You can view the page file directly in the browser, or, if you make a frame file, you can see the page file as part of the frame file (see “The Frame File” on page 645).

---

**The Frame File**

The frame file provides a simultaneous view of the body file and the contents file, the page file, or both. The following figure illustrates how a frame that references both the contents and page files looks (in part) to an ASCII editor. The SRC attribute identifies a file to display in the browser. ODS constructs the value for the SRC attribute the same way that it constructs the value for an HREF attribute in a page or contents file (see Schematic of an HTML Frame File on page 646).
Figure A2.5  Schematic of an HTML Frame File

HTML Frame File: pop-frame.htm

Browser View of HTML Frame File
Display A2.1 on page 647 shows the same frame file viewed from a browser.
**Display A2.1**  Browser View of HTML Frame File
Using a z/OS UNIX System Services HFS Directory for HTML Output

```/* Specify the files to create for the HTML output. */
/* The PATH= option specifies the location for all */
/* the HTML files. The URL= suboption prevents */
/* information from PATH= from appearing in the */
/* links and references that ODS creates. The URLs */
/* will be the same as the file specifications. */
ods html body='odsexample-body.htm'
    contents='odsexample-contents.htm'
    page='odsexample-page.htm'
    frame='odsexample-frame.htm'
    path='~'(url=none);
```

Using a z/OS PDSE for EBCDIC HTML Output

```/* Allocate a PDSE for the HTML Output. */
filename pdsehtml '.example.htm'
    dsntype=library dsorg=po
    disp=(new, catlg, delete);

/* Specify the files to create for the HTML output. */
/* These files are PDSE members. */
/* The PATH= option specifies the location for all */
/* the HTML files. The URL= suboption prevents */
/* information from PATH= from appearing in the */
/* links and references that ODS creates. The URLs */
/* will be the same as the file specifications. */
/* The RS= option creates HTML that you can work */
/* with in an editor and use on a z/OS web server. */
```
ods html body='odsexb' contents='odsexc' page='odsexp' frame='odsexf' path='.example.htm'(url=none) rs=none;

Using a z/OS PDSE for ASCII HTML Output

/* Allocate a PDSE for the HTML Output. */
filename pdsehtml '.example.htm'
   dsntype=library dsorg=po disp=(new, catlg, delete);

/* Specify the files to create for the HTML output. */
/* These files are PDSE members. */
/* The URL= suboption in the HTML-file */
/* specifications provides a URL that will be valid */
/* after the PDSE members have been moved to an */
/* ASCII file system. When the files are */
/* transferred, they must retain their member names */
/* and have the "htm" extension added in order for */
/* these URLs to be correct. */
/* The PATH= option specifies the location for all */
/* the HTML files. The URL= suboption in the PATH= */
/* option prevents information from PATH= from */
/* appearing in the links and references that ODS */
/* creates because it will not be a valid URL for */
/* the ASCII file system. */
/* The TRANTAB= option creates ASCII HTML that */
/* you can send to an ASCII-based web server. */

ods html body='odsexb' (url='odsexb.htm') contents='odsexc' (url='odsexc.htm') page='odsexp' (url='odsexp.htm') frame='odsexf' path='.example.htm'(url=none) trantab=ascii;

Note: Use a binary transfer to move the files to the web server. △
## Style Elements and Their Inheritances

The following table lists all the style elements in the default HTML and markup languages style definition. The table provides a brief description of each style element and indicates the style elements from which it inherits its attributes. An abstract style element is one that is not used to generate any style element but provides a parent for one or more style elements to inherit.

### Table A4.1  Style Elements That Are Available in the Default HTML, Printer Family, and Markup Languages Style Definition

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
<th>Valid Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>fonts</td>
<td>Establishes a list of fonts</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>color_list</td>
<td>Establishes a list of color names and their RGB values</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>colors</td>
<td>Associates parts of SAS output with colors from color_list</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>html</td>
<td>Provides HTML for specific parts of the output</td>
<td></td>
<td>HTML</td>
</tr>
<tr>
<td>text</td>
<td>Provides text for specific parts of the output</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>container</td>
<td>Abstract: provides a basis for all containers</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>index</td>
<td>Abstract: provides a basis for the contents and page files</td>
<td></td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>indexprocname</td>
<td>Inserts the procedure name in the body file</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>contentprocname</td>
<td>Inserts the procedure name in the contents file</td>
<td>indexprocname</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td>contentproclabel</td>
<td>Inserts the procedure label in the contents file</td>
<td>contentprocname</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>pagesprocname</td>
<td>Inserts the procedure name in the page file</td>
<td>indexprocname</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>pagesproclabel</td>
<td>Inserts the procedure label in the page file</td>
<td>pagesprocname</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>indexaction</td>
<td>Abstract: determines what happens when the mouse is positioned over a folder</td>
<td>indexaction</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>folderaction</td>
<td>Determines what happens when the mouse is positioned over a folder</td>
<td>indexaction</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>itemaction</td>
<td>Determines what happens when the mouse is positioned over an item</td>
<td>indexaction</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>proclameaction</td>
<td>Determines what happens when the mouse is positioned over the procedure name in the table of contents</td>
<td>indexaction</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td>titleaction</td>
<td>Determines what happens when the mouse is positioned over a SAS title</td>
<td>indexaction</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>indextitle</td>
<td>Abstract: controls the title of the contents and page files</td>
<td>index</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>contenttitle</td>
<td>Inserts the title in the contents file</td>
<td>indextitle</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
<td>-----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>pagestitle</td>
<td>Inserts the title in the page file</td>
<td>indextitle</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>index</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>document</td>
<td>Abstract: controls the various output files</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>body</td>
<td>Generates the body file</td>
<td>document</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>frame</td>
<td>Generates the frame file</td>
<td>document</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>contents</td>
<td>Generates the contents file</td>
<td>document</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>pages</td>
<td>Generates the page file</td>
<td>document</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>Abstract: controls the contents of date fields</td>
<td>container</td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>bodydate</td>
<td>Inserts the date field in the body file</td>
<td>date</td>
<td>RTF, Printer family</td>
</tr>
<tr>
<td>contentsdate</td>
<td>Inserts the date field in the contents file</td>
<td>date</td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>pagesdate</td>
<td>Inserts the date field in the page file</td>
<td>date</td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>indexitem</td>
<td>Abstract: controls the items in the contents and page files</td>
<td>container</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>contentfolder</td>
<td>Controls the generic folder definition in the contents file</td>
<td>indexitem</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td>bycontentfolder</td>
<td>Controls the byline folder definition in the contents file</td>
<td>contentfolder</td>
<td>HTML, MARKUP</td>
</tr>
<tr>
<td>contentitem</td>
<td>Inserts the lowest level of the hierarchy in a contents file</td>
<td>indexitem</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td>pagesitem</td>
<td>Inserts the lowest level of the hierarchy in a page file</td>
<td>indexitem</td>
<td>HTML, MARKUP, PDF</td>
</tr>
<tr>
<td>systitleandfootercontainer</td>
<td>Inserts the container for system headers and footers</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>titleandnotecontainer</td>
<td>Generates the container for titles and notes that the procedure provides</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>titlesandfooters</td>
<td>Abstract: controls the text of the system titles and footers</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>systemtitle</td>
<td>Generates the text of system titles</td>
<td>titlesandfooters container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>systemfooter</td>
<td>Generates the text of system footers</td>
<td>titlesandfooters container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>pageno</td>
<td>Generates the text of the page number</td>
<td>titlesandfooters container</td>
<td>RTF, Printer family</td>
</tr>
<tr>
<td>byline</td>
<td>Generates the text of the byline.</td>
<td>titlesandfooters container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>proctitle</td>
<td>Inserts the text of titles that the procedure provides</td>
<td>titlesandfooters container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>proctitlefixed</td>
<td>Inserts the text of titles that the procedure provides with a fixed font</td>
<td>proctitle</td>
<td>HTML, MARKUP, RTF, PCL, PDF</td>
</tr>
<tr>
<td>bylinecontainer</td>
<td>Generates the container for the byline</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>output</td>
<td>Abstract: controls basic presentation of the output</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>table</td>
<td>Generates output that is a table</td>
<td>output</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>batch</td>
<td>Generates output (for example, lineprinter plots and calendars) that requires a fixed font</td>
<td>output</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>note</td>
<td>Abstract: controls the container for the text that precedes notes, warning, and errors from SAS</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>notebanner</td>
<td>Generates the text that precedes the contents of a note</td>
<td>note</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>notecontent</td>
<td>Generates the contents of a note</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>notecontentfixed</td>
<td>Generates the contents of a note with a fixed font</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>warnbanner</td>
<td>Generates the text that precedes the contents of a warning</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>warncontent</td>
<td>Generates the contents of a warning</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>warncontentfixed</td>
<td>Generates the contents of a warning with a fixed font</td>
<td>warncontent container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>errorbanner</td>
<td>Generates the text that precedes the contents of an error</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>errorcontent</td>
<td>Generates the contents of an error</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>errorcontentfixed</td>
<td>Generates the contents of an error with a fixed font</td>
<td>errorcontent container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>fatalbanner</td>
<td>Generates the text that precedes the contents of a fatal error</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>fatalcontent</td>
<td>Generates the contents of a fatal error</td>
<td>note container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>fatalcontentfixed</td>
<td>Generates the contents of a fatal error with a fixed font</td>
<td>fatalcontent container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>cell</td>
<td>Abstract: controls table cells</td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>data</td>
<td>Generates the data in table cells</td>
<td>cell container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>datafixed</td>
<td>Generates the data in table cells with a fixed font</td>
<td>data container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>dataempty</td>
<td>Generates empty table cells</td>
<td>data container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>dataemphasis</td>
<td>Generates data in table cells with emphasis</td>
<td>data cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>dataemphasisfixed</td>
<td>Generates data in table cells with emphasis and a fixed font</td>
<td>data cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>datastrong</td>
<td>Generates data in cells with more emphasis than dataemphasis</td>
<td>data cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>datastrongfixed</td>
<td>Generates data in table cells with more emphasis than dataemphasis and with a fixed font</td>
<td>data cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>headersandfooters</td>
<td>Abstract: controls table headers and footers</td>
<td>cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>header</td>
<td>Generates the headers of a table</td>
<td>headersandfooters cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>headerfixed</td>
<td>Generates the headers of a table with a fixed font</td>
<td>header cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>headerempty</td>
<td>Generates empty table headers</td>
<td>header cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>headeremphasis</td>
<td>Generates the headers of a table with emphasis</td>
<td>header cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>headeremphasisfixed</td>
<td>Generates the headers of a table with emphasis and with a fixed font</td>
<td>header cell container</td>
<td>HTML, MARKUP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>headerstrong</td>
<td>Generates the headers of a table with more emphasis than <strong>headeremphasis</strong></td>
<td>header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>headerstrongfixed</td>
<td>Generates the headers of a table with more emphasis than <strong>headeremphasis</strong> and with a fixed font</td>
<td>header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowheader</td>
<td>Generates row headers</td>
<td>header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowheaderfixed</td>
<td>Generates row headers with a fixed font</td>
<td>rowheader, header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowheaderempty</td>
<td>Generates empty row headers</td>
<td>rowheader, header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowheaderemphasis</td>
<td>Generates row headers with emphasis</td>
<td>rowheader, header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowheaderemphasisfixed</td>
<td>Generates row headers with emphasis and with a fixed font</td>
<td>rowheaderemphasis, rowheader, header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowheaderstrong</td>
<td>Generates row headers with more emphasis than <strong>rowheaderemphasis</strong></td>
<td>rowheader, header, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>rowheaderstrongfixed</td>
<td>Generates row headers with more emphasis than rowheaderemphasis and with a fixed font</td>
<td>rowheaderstrong rowheader header headersandfooters cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footer</td>
<td>Generates the footers of a table</td>
<td>headersandfooters cell</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footerfixed</td>
<td>Generates the footers of a table with a fixed font</td>
<td>footer headersandfooters cell</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footerempty</td>
<td>Generates empty table footers</td>
<td>footer headersandfooters cell</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footeremphasis</td>
<td>Generates the footers of a table with emphasis</td>
<td>footer headersandfooters cell</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footeremphasisonfixed</td>
<td>Generates the footers of a table with emphasis and with a fixed font</td>
<td>footeremphasis footer headersandfooters cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footerstrong</td>
<td>Generates the footers of a table with more emphasis than footeremphasis</td>
<td>footer headersandfooters cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>footerstrongfixed</td>
<td>Generates the footers of a table with more emphasis than footeremphasis and with a fixed font</td>
<td>footerstrong footer headersandfooters cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>rowfooter</td>
<td>Generates row footers</td>
<td>footer headersandfooters cell</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>rowfooterfixed</td>
<td>Generates row footers with a fixed font</td>
<td>rowfooter</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>footer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>rowfootereempty</td>
<td>Generates empty row footers</td>
<td>rowfooter</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>footer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>rowfooterephasis</td>
<td>Generates row footers with emphasis</td>
<td>rowfooter</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>footer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>rowfooterephasisfixed</td>
<td>Generates row footers with emphasis and with a fixed font</td>
<td>rowfooterephasis</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rowfooter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>footer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>rowfooterstrong</td>
<td>Generates row footers with more emphasis than <em>rowfooterephasis</em></td>
<td>rowfooter</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>footer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>rowfooterstrongfixed</td>
<td>Generates row footers with more emphasis than <em>rowfooterephasis</em> and with a fixed font</td>
<td>rowfooterstrong</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rowfooter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>footer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>caption</td>
<td>Abstract: controls the caption field in <strong>PROC TABULATE</strong></td>
<td>headersandfooters</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>beforecaption</td>
<td>Generates captions that precede the table</td>
<td>caption</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>headersandfooters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>aftercaption</td>
<td>Generates captions that follow the table</td>
<td>caption, headersandfooters, cell, container</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphWalls</td>
<td>Controls the wall color or image</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphCharts</td>
<td>Contains attributes that affect all charts in the graphics area</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphBackground</td>
<td>Controls the background color or image of the graph</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphData1...GraphData#</td>
<td>Contains attributes that affect graphics primitives that are used to represent data</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphLegendBackground</td>
<td>Controls the legend background color or image</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphFloor</td>
<td>Controls the floor color or image</td>
<td></td>
<td>HTML, MARKUP, DOCUMENT, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>DropShadowStyle</td>
<td>Changes the drop shadow attributes for the text in the graphs</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphLabelText</td>
<td>Specifies text attributes for axis labels and the title of the legend</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphValueText</td>
<td>Controls the axis text attributes on tick marks and entry text on the legend</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Determines grid line attributes for a graph</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphAxisLines</td>
<td>Specifies axis line and tick mark attributes for the graph</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>GraphBorderLines</td>
<td>Determines frame attributes around the axis area and legend</td>
<td></td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
<td>Valid Destinations</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>GraphOutlines</td>
<td>Controls line attributes around data perimeters including pie charts, bar charts, map regions, etc</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
<td></td>
</tr>
<tr>
<td>Graph</td>
<td>Controls the width and height of the entire graphics area</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
<td></td>
</tr>
<tr>
<td>TwoColorRamp</td>
<td>Determines the colors to use for maps with continuous response</td>
<td>HTML, MARKUP, RTF, PS, PDF, PCL</td>
<td></td>
</tr>
</tbody>
</table>

* Provided by TITLE and FOOTNOTE statements or by GTITLE and GFOOTNOTE statements in combination with the NOGTITLE and NOGFOOTNOTE options in the ODS HTML and ODS MARKUP statements.
Recommended Reading

Here is the recommended reading list for this title:

- Base SAS Procedures Guide
- SAS Language Reference: Concepts
- SAS Language Reference: Dictionary
- Step-by-Step Programming with Base SAS Software

The recommended reading list from Books By Users includes:

- The Little SAS Book: A Primer, Revised Second Edition
- Output Delivery System: The Basics

For a complete list of SAS publications, see the current SAS Publishing Catalog. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/pubs
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.
Index

A
ABBR= header attribute 400
ABSTRACT= style attribute 302
ACECLUS procedure
  ODS table names 438
Acrobat Distiller 166
ACRONYM= header attribute 400
actions
  ODS DOCUMENT statement 87
  ODS LISTING statement 107
  ODS MARKUP statement 135
  ODS PDF statement 153
  ODS PRINTER statement 160
  ODS RTF statement 180
ACTIVEFOOTN option
  REPLAY statement (DOCUMENT) 232
  ACTIVELINKCOLOR= style attribute 302
  ACTIVETITLE option
  REPLAY statement (DOCUMENT) 232
ActiveX devices
  CODEBASE file path 112
AFTER= option
  COPY statement (DOCUMENT) 216
  IMPORT statement (DOCUMENT) 220
  LINK statement (DOCUMENT) 221
  MAKE statement (DOCUMENT) 223
  MOVE statement (DOCUMENT) 224
  NOTE statement (DOCUMENT) 225
  OBPAGE statement (DOCUMENT) 229
aggregate storage location
definition 266
ALT= header attribute 400
ALT= table attribute 415
ANCHOR= option
  ODS MARKUP statement 110
  ODS PRINTER statement 161
  ODS RTF statement 181
anchor tags
  base name for 110, 181
  root name for 161
ANOVA procedure
  ODS table names 438
ANTIALIAS= option
  ODS GRAPHICS statement 93
APPEND option
  ODS PATH statement 150
  PATH statement (TEMPLATE) 277
  appending HTML files 101
  applets
    viewing HTML output 110
  ARCHIVE= option
  ODS MARKUP statement 110
  ARIMA procedure
  ODS table names 400
  AS option
    EDIT statement (TEMPLATE) 373
    ASIS= style attribute 302
    attribute suboptions
      FILE PRINT ODS statement 75
      ODS MARKUP statement 111
    AUTHOR= option
    ODS PRINTER statement 161
    ODS RTF statement 182
    automatic graphic capabilities
    AUTOREG procedure
    ODS table names 402
background color
  printing in text 161
  BACKGROUND= option
    ODS PRINTER statement 161
    BACKGROUND= style attribute 302
    BACKGROUNDIMAGE= style attribute 415
    BALANCE table attribute 303
    BASE= option
    ODS MARKUP statement 111
    ODS PRINTER statement 161
    ODS RTF statement 182
    base text 162
    HTML output 111
BEFORE= option
  COPY statement (DOCUMENT) 216
  IMPORT statement (DOCUMENT) 220
  LINK statement (DOCUMENT) 221
  MAKE statement (DOCUMENT) 223
  MOVE statement (DOCUMENT) 224
  NOTE statement (DOCUMENT) 225
  ODS MARKUP statement 111
  ODS PRINTER statement 161
  ODS RTF statement 182
  base text 162
  HTML output 111

Body files
  creating 118
  separate file per page of output 96
  BODYSIZE= style attribute 303
  BODYSCROLLBAR= style attribute 303
  BODYSCROLLBAR= style attribute 304
  BORDERCOLOR= style attribute 304
  BORDERCOLOR= style attribute 304
  BORDERCOLOR= style attribute 304
  BOTTOMMARGIN= style attribute 304
BREAK statement
  TEMPLATE procedure 561
  buffers
    number of columns in 304
    BULLETS= style attribute 304
    BY-groups
    DOCUMENT procedure and 233
    BY lines 235
    BY variable names 234
    BY variable values 234
    BYLINE= table attribute 416

C
CALENDAR procedure
  ODS table names 431
CALIS procedure
  ODS table names 440
CANCORR procedure
  ODS table names 444
CANDISC procedure
  ODS table names 446
cascading style sheets 105
CATALOG option
  ODS DOCUMENT statement 233
CATALOG procedure
  ODS table names 433
catalogs
  copying GSREGs to 88
CATMOD procedure
  ODS table names 447
celldatalarge style element 628
CELLHEIGHT= style attribute 305
CELLPADDING= style attribute 305
CELLSPACING= style attribute 305
CELLSTYLE AS statement, TEMPLATE procedure
column definitions 389
table definitions 422
CELLWIDTH= style attribute 305
CENTER table attribute 416
characterizer
META declaration for HTML output 112
CHARSET= option
ODS MARKUP statement 112
CHART procedure
ODS table names 431
CHOICE FORMAT= column attribute 380
CHOOSE_FORMAT= column attribute 380
CHART procedure 382
CLUSTER procedure
CLOSE statement
CLOSE action
CLEAR action
ODS OUTPUT statement 135
CLOSE action
ODS DOCUMENT statement 89
ODS LISTING statement 107
ODS MARKUP statement 109
ODS OUTPUT statement 135
ODS PRINTER statement 160
ODS RTF statement 181
CLOSE statement
TEMPLATE procedure 562
CLUSTER procedure
ODS table names 448
CODEBASE file path 112
CODEBASE= option
ODS MARKUP statement 112
COLOR= option
ODS PRINTER statement 163
COLORTAGSET tagset 121
CLASSLEVELS= table attribute 416
COLORS= option
ODS PRINTER statement 163
COLORLATEX tagset 121
colors
ODS PRINTER statement 163
COL SPACE MAX= table attribute 416
COL SPACE MIN= table attribute 416
column attributes 376
values from data component 78
column definitions attributes 376
creating 375
editing 273
for multiple variables 72
header definitions in column pointer controls
ODS 62
COLUMN statement
TEMPLATE procedure 423
columns
assigning attributes to cell styles 389
changing without redefining for data components 70
formats for justification 514
labels for notes about number in buffers 394
number in data components 83
ODS PRINTER statement 163
ODS RTF statement 182
specifying 73
symbol declared as 423
COLUMNS= option
ODS RTF statement 182
COLUMN= suboption
FILE PRINT ODS statement comma-delimited output 65
COMPARE procedure
ODS table names compatibility
ODS documents COMPRESS= option
ODS PRINTER statement compression
PDF files 163
COMPUTE AS statement
TEMPLATE procedure
computed columns 390
CONTENTPOSITION= style attribute 306
CONTENTSIZE= style attribute 306
CONTENTSCROLLBAR= style attribute 306
CONTENTS table attribute 416
CONTENTPOSITION= style attribute 306
CONTENTSIZE= style attribute 306
CONTENTSCROLLBAR= style attribute 306
CONTROL= table attribute 416
COPY statement
DOCUMENT procedure 216
COPYRIGHT= tagset attribute 554
COPYRIGHT= tagset attribute 554
CORR procedure
ODS table names 432
CORRESP procedure
ODS table names 449
COSAN model 440
CSV tagset 121
CSVALL destination 85
CSVALL tagset 122
csvbyline tagset 122
csvbyline tagset current document closing definition 219
current file location creating text strings in importing data sets to
importing GRSEGs to current path definition 212
customized output 34
number of columns in 83
data panels data sets
combined output data sets 138
creating with/without MATCH_ALL option
from output objects 135
from similar output objects 142
merging dissimilar output objects into 137
DATA step
column definitions for multiple variables 72
ODS and 39
ODS enhanced features in ODS examples 41
ODS reports with 40
DATA step statements
ODS 61
DATA_FORMAT OVERRIDE column attribute 390
DATA_FORMAT OVERRIDE table attribute 417
DATANAME= column attribute 381
DATAPANEL= option
ODS LISTING statement 108
DATASETS procedure
ODS table names 433
decimal point in numeric columns 66
DEFAULT EVENT= tagset attribute 555
DEFINE COLUMN statement
TEMPLETE procedure 375
DEFINE EVENT statement, TEMPLATE procedure 559
event attributes 560
event statement conditions 571
event variables 572
DEFINE FOOTER statement
TEMPLATE procedure 395
DEFINE HEADER statement
TEMPLATE procedure 391
DEFINE statement
TEMPLATE procedure 425
DEFINE STYLE statements
TEMPLATE procedure 288
DEFINE TABLE statement
TEMPLATE procedure 410
vs. EDIT statement 520
DEFINE TAGSET statement
TEMPLATE procedure 553
DEFINE EVENT statement
TEMPLATE procedure 558
DEF_SPLIT column attribute 381
DEF_SPLIT header attribute 400
DELETE option
OBPAGE statement (DOCUMENT) 229
DELETE statement
DOCUMENT procedure 217
TEMPLATE procedure 273
delimiters in tags 605
DELSTREAM statement
TEMPLATE procedure 562
DEST= option
REPLAY statement (DOCUMENT) 232
<table>
<thead>
<tr>
<th>ODS table names</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACECLUS procedure</td>
<td>438</td>
</tr>
<tr>
<td>ANOVA procedure</td>
<td>438</td>
</tr>
<tr>
<td>ARIMA procedure</td>
<td>490</td>
</tr>
<tr>
<td>AUTOREG procedure</td>
<td>492</td>
</tr>
<tr>
<td>Base SAS procedures</td>
<td>439</td>
</tr>
<tr>
<td>CALENDAR procedure</td>
<td>431</td>
</tr>
<tr>
<td>CALIS procedure</td>
<td>440</td>
</tr>
<tr>
<td>CANCORR procedure</td>
<td>444</td>
</tr>
<tr>
<td>CANDISC procedure</td>
<td>446</td>
</tr>
<tr>
<td>CATALOG procedure</td>
<td>431</td>
</tr>
<tr>
<td>CATMOD procedure</td>
<td>447</td>
</tr>
<tr>
<td>CHART procedure</td>
<td>431</td>
</tr>
<tr>
<td>CLUSTER procedure</td>
<td>448</td>
</tr>
<tr>
<td>COMPARE procedure</td>
<td>431</td>
</tr>
<tr>
<td>CONTENTS procedure</td>
<td>433</td>
</tr>
<tr>
<td>CORR procedure</td>
<td>432</td>
</tr>
<tr>
<td>CORRESP procedure</td>
<td>449</td>
</tr>
<tr>
<td>DATASETS procedure</td>
<td>433</td>
</tr>
<tr>
<td>DISCRIM procedure</td>
<td>450</td>
</tr>
<tr>
<td>ENTROPY procedure</td>
<td>493</td>
</tr>
<tr>
<td>FACTOR procedure</td>
<td>452</td>
</tr>
<tr>
<td>FASTCLUS procedure</td>
<td>454</td>
</tr>
<tr>
<td>FREQ procedure</td>
<td>455</td>
</tr>
<tr>
<td>GAM procedure</td>
<td>456</td>
</tr>
<tr>
<td>GENMOD procedure</td>
<td>458</td>
</tr>
<tr>
<td>GLMMOD procedure</td>
<td>461</td>
</tr>
<tr>
<td>INBREED procedure</td>
<td>461</td>
</tr>
<tr>
<td>KDE procedure</td>
<td>462</td>
</tr>
<tr>
<td>LATTICE procedure</td>
<td>462</td>
</tr>
<tr>
<td>LIFEREG procedure</td>
<td>462</td>
</tr>
<tr>
<td>LIFETEST procedure</td>
<td>463</td>
</tr>
<tr>
<td>LOAN procedure</td>
<td>494</td>
</tr>
<tr>
<td>LOESS procedure</td>
<td>463</td>
</tr>
<tr>
<td>LOGISTIC procedure</td>
<td>465</td>
</tr>
<tr>
<td>MDC procedure</td>
<td>494</td>
</tr>
<tr>
<td>MDS procedure</td>
<td>468</td>
</tr>
<tr>
<td>MEANS procedure</td>
<td>435</td>
</tr>
<tr>
<td>MI procedure</td>
<td>468</td>
</tr>
<tr>
<td>MIANALYZE procedure</td>
<td>469</td>
</tr>
<tr>
<td>MODECLUS procedure</td>
<td>470</td>
</tr>
<tr>
<td>MODEL procedure</td>
<td>495</td>
</tr>
<tr>
<td>MULTTEST procedure</td>
<td>471</td>
</tr>
<tr>
<td>NESTED procedure</td>
<td>471</td>
</tr>
<tr>
<td>NLIN procedure</td>
<td>471</td>
</tr>
<tr>
<td>NLMIXED procedure</td>
<td>472</td>
</tr>
<tr>
<td>NPAR1WAY procedure</td>
<td>473</td>
</tr>
<tr>
<td>orthoreg procedure</td>
<td>474</td>
</tr>
<tr>
<td>PQLREG procedure</td>
<td>477</td>
</tr>
<tr>
<td>PLAN procedure</td>
<td>478</td>
</tr>
<tr>
<td>PLOT procedure</td>
<td>479</td>
</tr>
<tr>
<td>PLS procedure</td>
<td>476</td>
</tr>
<tr>
<td>PPHREG procedure</td>
<td>475</td>
</tr>
<tr>
<td>PRINCOMP procedure</td>
<td>477</td>
</tr>
<tr>
<td>PRINQUAL procedure</td>
<td>478</td>
</tr>
<tr>
<td>PROBIT procedure</td>
<td>478</td>
</tr>
<tr>
<td>REG procedure</td>
<td>479</td>
</tr>
<tr>
<td>REPORT procedure</td>
<td>436</td>
</tr>
<tr>
<td>ROBUSTREG procedure</td>
<td>480</td>
</tr>
<tr>
<td>RSREG procedure</td>
<td>482</td>
</tr>
<tr>
<td>SAS/ETS procedures</td>
<td>490</td>
</tr>
<tr>
<td>SAS/STAT procedures</td>
<td>437</td>
</tr>
<tr>
<td>SIMLIN procedure</td>
<td>498</td>
</tr>
<tr>
<td>SPECTRA procedure</td>
<td>496</td>
</tr>
<tr>
<td>SQL procedure</td>
<td>456</td>
</tr>
<tr>
<td>STATESPACE procedure</td>
<td>499</td>
</tr>
<tr>
<td>STDIZE procedure</td>
<td>482</td>
</tr>
<tr>
<td>STEPDISC procedure</td>
<td>483</td>
</tr>
<tr>
<td>SUMMARY procedure</td>
<td>485</td>
</tr>
<tr>
<td>SURVEYMEANS procedure</td>
<td>483</td>
</tr>
<tr>
<td>SURVEYREG procedure</td>
<td>484</td>
</tr>
<tr>
<td>SURVEYSELECT procedure</td>
<td>485</td>
</tr>
<tr>
<td>SYSLIN procedure</td>
<td>500</td>
</tr>
<tr>
<td>TABULATE procedure</td>
<td>436</td>
</tr>
<tr>
<td>TIMEPLOT procedure</td>
<td>437</td>
</tr>
<tr>
<td>TIMESERIES procedure</td>
<td>501</td>
</tr>
<tr>
<td>TPSPLINE procedure</td>
<td>487</td>
</tr>
<tr>
<td>TRANSREG procedure</td>
<td>487</td>
</tr>
<tr>
<td>TREE procedure</td>
<td>488</td>
</tr>
<tr>
<td>TTEST procedure</td>
<td>489</td>
</tr>
<tr>
<td>UNIVARIATE procedure</td>
<td>437</td>
</tr>
<tr>
<td>VARCLUS procedure</td>
<td>440</td>
</tr>
<tr>
<td>VARCOMP procedure</td>
<td>449</td>
</tr>
<tr>
<td>VARMAX procedure</td>
<td>502</td>
</tr>
<tr>
<td>X11 procedure</td>
<td>506</td>
</tr>
<tr>
<td>X12 procedure</td>
<td>510</td>
</tr>
<tr>
<td>ODS TRACE statement</td>
<td>198</td>
</tr>
<tr>
<td>contents of trace record</td>
<td>198</td>
</tr>
<tr>
<td>example 200</td>
<td>198</td>
</tr>
<tr>
<td>LABEL= option</td>
<td>35</td>
</tr>
<tr>
<td>purpose 135</td>
<td>136</td>
</tr>
<tr>
<td>specifying output objects</td>
<td>199</td>
</tr>
<tr>
<td>ODS USEG0PT statement</td>
<td>203</td>
</tr>
<tr>
<td>ODS VERIFY statement</td>
<td>205</td>
</tr>
<tr>
<td>ODS WML statement</td>
<td>206</td>
</tr>
<tr>
<td>ODSSTYLE tagset</td>
<td>123</td>
</tr>
<tr>
<td>ODSXRPCS tagset</td>
<td>123</td>
</tr>
<tr>
<td>OPEN statement</td>
<td>563</td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>563</td>
</tr>
<tr>
<td>OPERATOR= option</td>
<td>184</td>
</tr>
<tr>
<td>ODS RTF statement</td>
<td>185</td>
</tr>
<tr>
<td>OPTIONAL column attribute</td>
<td>385</td>
</tr>
<tr>
<td>ORDER= option</td>
<td>385</td>
</tr>
<tr>
<td>LIST statement (DOCUMENT)</td>
<td>222</td>
</tr>
<tr>
<td>ORDER_DATA table attribute</td>
<td>170</td>
</tr>
<tr>
<td>ORTHOREG procedure</td>
<td>419</td>
</tr>
<tr>
<td>ODS table names</td>
<td>474</td>
</tr>
<tr>
<td>OS/390 printing output directly to printer</td>
<td>170</td>
</tr>
<tr>
<td>Output Control statements</td>
<td>62</td>
</tr>
<tr>
<td>OUTPUT destination</td>
<td>26</td>
</tr>
<tr>
<td>closing 135</td>
<td>135</td>
</tr>
<tr>
<td>definition 21</td>
<td>21</td>
</tr>
<tr>
<td>exclusion lists</td>
<td>135</td>
</tr>
<tr>
<td>selection lists</td>
<td>135</td>
</tr>
<tr>
<td>output objects</td>
<td>237</td>
</tr>
<tr>
<td>attributes 237</td>
<td>135</td>
</tr>
<tr>
<td>creating 237</td>
<td>135</td>
</tr>
<tr>
<td>customized output for data sets from definition 21</td>
<td>237</td>
</tr>
<tr>
<td>determining destinations for excluding from ODS destinations 55 30 135 90 228 87 72 229 198 229 137 321</td>
<td>228</td>
</tr>
<tr>
<td>footnotes 228</td>
<td>87</td>
</tr>
<tr>
<td>librefs 76</td>
<td>72</td>
</tr>
<tr>
<td>listing output</td>
<td>198</td>
</tr>
<tr>
<td>merging dissimilar objects into data set names for page breaks records in log 198 229 137</td>
<td>137</td>
</tr>
<tr>
<td>renaming 231</td>
<td>198</td>
</tr>
<tr>
<td>number of lines for 68</td>
<td>68</td>
</tr>
<tr>
<td>output pointer</td>
<td>221</td>
</tr>
<tr>
<td>symbolic links to/from tracing 198</td>
<td>221</td>
</tr>
<tr>
<td>output pointer</td>
<td>221</td>
</tr>
<tr>
<td>OutputHEIGHT= style attribute</td>
<td>314</td>
</tr>
<tr>
<td>OUTPUT_TYPE= tagset attribute</td>
<td>556</td>
</tr>
<tr>
<td>OUTPUTWIDTH= style attribute</td>
<td>315</td>
</tr>
<tr>
<td>overflow-control option</td>
<td>68</td>
</tr>
<tr>
<td>FILE PRINT ODS statement</td>
<td>68</td>
</tr>
<tr>
<td>OVERHANGFACTOR= style attribute</td>
<td>315</td>
</tr>
<tr>
<td>OVERLINE column attribute</td>
<td>385</td>
</tr>
<tr>
<td>OVERLINE header attribute</td>
<td>402</td>
</tr>
<tr>
<td>OVERLINE table attribute</td>
<td>419</td>
</tr>
</tbody>
</table>

### P

- page breaks 168
- output objects 229
- RTF output 185
- splitting tables at 184
- PageBreakHTML= style attribute 315
- PANELS= table attribute 419
- PANEL_SPACE= table attribute 419
- PARAMETERS= option 118
- ODS MARKUP statement 118
- PARENT= column attribute 385
- PARENT= header attribute 402
- PARENT= option 119
- DEFINE STYLE statements (TEMPLATE) 289
- PARENT= table attribute 419
- PARENT= tagset attribute 556
- PATH= option 119
- ODS MARKUP statement 119
- PATH statement 276
- TEMPLATE procedure 276
- paths 276
- definition 213
- PCL destination 151
- closing 152
- opening 152
- PCL files 151
- PCL option 155
- ODS PRINTER statement 165
- PCL output 163
- PDF destination 153
- closing 155
- opening 155
- opening multiple instances 155
- PDF files 166
- adding notes 166
- compressing list of bookmarks 162
- PDF option 166
- ODS PRINTER statement 166
- PDF output 166
- sample 18
- PDFMARK option 166
- ODS PRINTER option 166
- ODS PRINTER statement 166
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDLREG procedure</td>
<td>497</td>
</tr>
<tr>
<td>PERSIST option</td>
<td>94</td>
</tr>
<tr>
<td>ODS GRAPHICS statement</td>
<td></td>
</tr>
<tr>
<td>procedure labels</td>
<td>176</td>
</tr>
<tr>
<td>procedure statements</td>
<td>82</td>
</tr>
<tr>
<td>procedures</td>
<td></td>
</tr>
<tr>
<td>creating data sets from output objects</td>
<td></td>
</tr>
<tr>
<td>editing table definitions</td>
<td>515</td>
</tr>
<tr>
<td>ODS documents and Base procedures</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td></td>
</tr>
<tr>
<td>Base SAS</td>
<td></td>
</tr>
<tr>
<td>SAS/ETS</td>
<td></td>
</tr>
<tr>
<td>SAS/STAT</td>
<td></td>
</tr>
<tr>
<td>style definitions with</td>
<td>31</td>
</tr>
<tr>
<td>title in output</td>
<td></td>
</tr>
<tr>
<td>Properties window</td>
<td>243</td>
</tr>
<tr>
<td>PROTECTSPECIALCHARACTERS= style attribute</td>
<td></td>
</tr>
<tr>
<td>PS destination</td>
<td>177</td>
</tr>
<tr>
<td>closing</td>
<td></td>
</tr>
<tr>
<td>opening</td>
<td>179</td>
</tr>
<tr>
<td>PS format</td>
<td>95</td>
</tr>
<tr>
<td>PS option</td>
<td></td>
</tr>
<tr>
<td>ODS PRINTER statement</td>
<td>167</td>
</tr>
<tr>
<td>PURE_STYLE= event attribute</td>
<td>560</td>
</tr>
<tr>
<td>PUT statement</td>
<td></td>
</tr>
<tr>
<td>ODS</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>564</td>
</tr>
<tr>
<td>PUTL statement</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>565</td>
</tr>
<tr>
<td>PUTLOG statement</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>565</td>
</tr>
<tr>
<td>PUTQ statement</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>565</td>
</tr>
<tr>
<td>PUTSTREAM statement</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>566</td>
</tr>
<tr>
<td>PUTVARS statement</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>566</td>
</tr>
<tr>
<td>PYX tagset</td>
<td>123</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>quotation marks</td>
<td></td>
</tr>
<tr>
<td>in event variables</td>
<td>566</td>
</tr>
<tr>
<td>in style variables</td>
<td>566</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RAM model</td>
<td>440</td>
</tr>
<tr>
<td>RECORD_SEPARATOR= option</td>
<td></td>
</tr>
<tr>
<td>ODS MARKUP statement</td>
<td>120</td>
</tr>
<tr>
<td>ODS RTF statement</td>
<td>184</td>
</tr>
<tr>
<td>references</td>
<td></td>
</tr>
<tr>
<td>See HTML references</td>
<td></td>
</tr>
<tr>
<td>REG procedure</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td>479</td>
</tr>
<tr>
<td>REGISTERED_TM= tagset attribute</td>
<td>557</td>
</tr>
<tr>
<td>registry</td>
<td></td>
</tr>
<tr>
<td>changing default HTML version setting</td>
<td></td>
</tr>
<tr>
<td>changing ODS destination default settings</td>
<td></td>
</tr>
<tr>
<td>ODS and</td>
<td>32</td>
</tr>
<tr>
<td>REMOVE option</td>
<td></td>
</tr>
<tr>
<td>ODS PATH statement</td>
<td>150</td>
</tr>
<tr>
<td>PATH statement (TEMPLATE)</td>
<td>277</td>
</tr>
<tr>
<td>RENAME statement</td>
<td></td>
</tr>
<tr>
<td>DOCUMENT procedure</td>
<td>231</td>
</tr>
<tr>
<td>REPEATE header attribute</td>
<td>403</td>
</tr>
<tr>
<td>REPLACE statement</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE procedure</td>
<td>290</td>
</tr>
<tr>
<td>replay</td>
<td></td>
</tr>
<tr>
<td>definition</td>
<td>213</td>
</tr>
<tr>
<td>REPLACE statement</td>
<td></td>
</tr>
<tr>
<td>DOCUMENT procedure</td>
<td></td>
</tr>
<tr>
<td>REPORT procedure</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td>436</td>
</tr>
<tr>
<td>style definitions with</td>
<td></td>
</tr>
<tr>
<td>title in output</td>
<td>51</td>
</tr>
<tr>
<td>REQUIRED_SPACE= table attribute</td>
<td>420</td>
</tr>
<tr>
<td>RESET option</td>
<td></td>
</tr>
<tr>
<td>ODS GRAPHICS statement</td>
<td>94</td>
</tr>
<tr>
<td>Results window</td>
<td>241</td>
</tr>
<tr>
<td>tracking ODS output</td>
<td></td>
</tr>
<tr>
<td>viewing entries</td>
<td>180</td>
</tr>
<tr>
<td>vs. Documents window</td>
<td>241</td>
</tr>
<tr>
<td>RIGHTMARGIN= style attribute</td>
<td>317</td>
</tr>
<tr>
<td>ROBUSTREG procedure</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td>481</td>
</tr>
<tr>
<td>root file location</td>
<td>213</td>
</tr>
<tr>
<td>definition</td>
<td></td>
</tr>
<tr>
<td>RSREG procedure</td>
<td>482</td>
</tr>
<tr>
<td>ODS table names</td>
<td></td>
</tr>
<tr>
<td>RTF destination</td>
<td>180</td>
</tr>
<tr>
<td>closing</td>
<td>181</td>
</tr>
<tr>
<td>definition</td>
<td>186</td>
</tr>
<tr>
<td>excluding output objects</td>
<td></td>
</tr>
<tr>
<td>opening</td>
<td>181</td>
</tr>
<tr>
<td>selecting output objects</td>
<td></td>
</tr>
<tr>
<td>writing selection/exclusion lists to log</td>
<td>181</td>
</tr>
<tr>
<td>RTF files</td>
<td></td>
</tr>
<tr>
<td>creating record separator</td>
<td>184</td>
</tr>
<tr>
<td>style definitions</td>
<td>186</td>
</tr>
<tr>
<td>time and date of SAS program</td>
<td>185</td>
</tr>
<tr>
<td>RTF output</td>
<td></td>
</tr>
<tr>
<td>creating</td>
<td></td>
</tr>
<tr>
<td>footnotes</td>
<td>185</td>
</tr>
<tr>
<td>graphics</td>
<td>187</td>
</tr>
<tr>
<td>inserting text</td>
<td>186</td>
</tr>
<tr>
<td>page breaks</td>
<td></td>
</tr>
<tr>
<td>sample</td>
<td>17</td>
</tr>
<tr>
<td>splitting tables at page breaks</td>
<td>185</td>
</tr>
<tr>
<td>titles</td>
<td>183</td>
</tr>
<tr>
<td>translation tables</td>
<td>186</td>
</tr>
<tr>
<td>RULES= style attribute</td>
<td>317</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SAS/ETS procedures</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td>490</td>
</tr>
<tr>
<td>SAS Explorer window</td>
<td></td>
</tr>
<tr>
<td>list of available styles</td>
<td>50</td>
</tr>
<tr>
<td>SAS formatted destinations</td>
<td>25</td>
</tr>
<tr>
<td>SAS formatted statements</td>
<td>26</td>
</tr>
<tr>
<td>SAS option</td>
<td></td>
</tr>
<tr>
<td>ODS PRINTER statement</td>
<td>168</td>
</tr>
<tr>
<td>SAS/STAT procedures</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td>437</td>
</tr>
<tr>
<td>SASDATE option</td>
<td></td>
</tr>
<tr>
<td>ODS RTF statement</td>
<td>185</td>
</tr>
<tr>
<td>SASEDOC argument</td>
<td></td>
</tr>
<tr>
<td>LIBNAME statement</td>
<td>76</td>
</tr>
<tr>
<td>SASEDOC engine</td>
<td></td>
</tr>
<tr>
<td>LIBNAME statement</td>
<td>76</td>
</tr>
<tr>
<td>SASFMAT tagset</td>
<td>76</td>
</tr>
<tr>
<td>SASXMISS tagset</td>
<td>123</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>UNBLOCK statement</td>
<td>VARCLUS procedure</td>
</tr>
<tr>
<td>DOCUMENT procedure</td>
<td>ODS table names</td>
</tr>
<tr>
<td>UNIFORM option</td>
<td>489</td>
</tr>
<tr>
<td>UNIX</td>
<td></td>
</tr>
<tr>
<td>UNDERLINE column attribute</td>
<td></td>
</tr>
<tr>
<td>UNHIDE statement</td>
<td></td>
</tr>
<tr>
<td>VJUST= column attribute</td>
<td></td>
</tr>
<tr>
<td>VJUST= header attribute</td>
<td></td>
</tr>
<tr>
<td>VJUST= style attribute</td>
<td></td>
</tr>
<tr>
<td>VMS</td>
<td></td>
</tr>
<tr>
<td>WATERMARK= style attribute</td>
<td></td>
</tr>
<tr>
<td>VARCOMP procedure</td>
<td></td>
</tr>
<tr>
<td>ODS table names</td>
<td></td>
</tr>
<tr>
<td>VARIABLES= suboption</td>
<td></td>
</tr>
<tr>
<td>V JUST= column attribute</td>
<td></td>
</tr>
<tr>
<td>VJUST= header attribute</td>
<td></td>
</tr>
<tr>
<td>WAP (Wireless Application Protocol)</td>
<td></td>
</tr>
<tr>
<td>WML destination</td>
<td></td>
</tr>
<tr>
<td>WML tagset</td>
<td>125</td>
</tr>
<tr>
<td>WMLIST tagset</td>
<td></td>
</tr>
<tr>
<td>WRAP table attribute</td>
<td></td>
</tr>
<tr>
<td>XML files</td>
<td></td>
</tr>
<tr>
<td>XML tagset</td>
<td>122</td>
</tr>
<tr>
<td>XML output</td>
<td></td>
</tr>
<tr>
<td>X11 procedure</td>
<td></td>
</tr>
<tr>
<td>X12 procedure</td>
<td></td>
</tr>
<tr>
<td>X11 procedure</td>
<td></td>
</tr>
<tr>
<td>XML files</td>
<td></td>
</tr>
<tr>
<td>XENT statement</td>
<td></td>
</tr>
</tbody>
</table>

INDEX 675

<table>
<thead>
<tr>
<th>UNDERLINE column attribute</th>
<th>UNDERLINE header attribute</th>
</tr>
</thead>
</table>

| TEMPLATE procedure       | UNDERLINE table attribute |
| DOCUMENT procedure       | UNHIDE statement          |
| UNIFORM option           | UNIVERSITY procedure      |
| UNIX                     |                           |
| USE_FORMAT_DEFAULTS table attribute |               |
| USE_NAME table attribute  |                           |
| USER-DEFINED TAGSETS      |                           |
| USER-DEFINED VARIABLES    |                           |
| VJUST= column attribute   |                           |
| VJUST= header attribute   |                           |
| VJUST= style attribute    |                           |
| VMS                      |                           |
| WATERMARK= style attribute |                          |
| WAP (Wireless Application Protocol) |             |
| WML destination          |                           |
| WML tagset               | 125                       |
| WMLIST tagset            |                           |
| WRAP table attribute      |                           |
| XML files                 |                           |
| XML tagset                | 122                       |

| WIDTH= column attribute   | WIDTH= header attribute   |
| WIDTH_MAX= column attribute |                           |
| Windows                   |                           |
| ODS PRINTER statement     |                           |
| printing output directly to printer | \[170\] |
| Wireless Application Protocol (WAP) | \[206\] |
| Wireless Markup Language DTD | \[206\] |

INDEX 675
Your Turn

If you have comments or suggestions about SAS® 9.1 Output Delivery System: User's Guide, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: yourturn@sas.com

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: suggest@sas.com