

CHAPTER

4

Using the Output Delivery System in the DATA Step

<i>Overview</i>	105
<i>Statement Syntax for Use with ODS</i>	105
<i>FILE Statement for ODS</i>	106
<i>_ODS_ in the PUT Statement</i>	112
<i>Examples</i>	114
<i>Example 1: Using the Default Table Definition</i>	114
<i>Example 2: Selecting Variables for the Data Component</i>	117
<i>Example 3: Specifying Attributes for a Column</i>	121
<i>Example 4: Using a Simple User-Defined Definition</i>	126

Overview

Within the DATA step itself, the ODS option in the FILE statement and the _ODS_ option in the PUT statement provide connections with the Output Delivery System. You use both of these connections to route the results of a DATA step to ODS. By default, when the DATA step uses ODS, ODS writes output objects to all open ODS destinations and places links to them in the Results folder. You can use global ODS statements (see Chapter 3, “The ODS Statements,” on page 47) to write to other ODS destinations.

Statement Syntax for Use with ODS

FILE PRINT <ODS<=(*ODS-option(s)*)>><*overflow-control*><N=*number*>;

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

The FILE and PUT statements interact in the following way:

- The ODS option in the FILE statement defines the structure of the data component that holds the results of the DATA step.
- The _ODS_ option in the PUT statement writes values (as specified by the ODS option in the FILE statement) into a special buffer. This buffer is written to the data component.
- The ODS option in the FILE statement binds the data component to a table definition to produce an output object. ODS sends this object to all open ODS destinations, each of which formats the object appropriately.

The ODS destinations are controlled by the global ODS statements. You can use an existing table definition or create your own with PROC TEMPLATE (see Chapter 5, “The TEMPLATE Procedure,” on page 131).

This section provides information on using the Output Delivery System from the DATA step. For conceptual information on how ODS works, see Chapter 2, “Basic Concepts of the Output Delivery System,” on page 21. For information on the ODS statements, see Chapter 3, “The ODS Statements,” on page 47.

FILE Statement for ODS

Defines the structure of the data component that holds the results of the DATA step and binds that component to a table definition to produce an output object. ODS sends this object to all open ODS destinations, each of which formats the object appropriately. Also controls what happens when the PUT statement tries to write past the end of a line.

Requirements: If you use the ODS option, you must use the fileref PRINT in the FILE statement.

Restriction: The DELIMITER= and DSD options have no effect on the ODS option. The FOOTNOTES | NOFOOTNOTES, LINESIZE, PAGESIZE, and TITLES | NOTITLES options have an effect only on the listing destination. You cannot use _FILE_=, FILEVAR=, HEADER=, or PAD with the ODS option.

FILE PRINT <ODS<=(*ODS-suboption(s)*)>><*overflow-control*><N=*number*> ;

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

File Statement Arguments and Options

The arguments and options listed here can be specified in any order in the FILE statement.

N=*number*

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

ODS<=(*ODS-suboptions*)>

Defines the structure of the data component that holds the results of the DATA step and binds that component to a table definition to produce an output object. ODS sends this object to all open ODS destinations, each of which formats the object appropriately. For information on the ODS-suboptions, see “ODS Suboptions” on page 107.

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

Without suboptions, the default table definition uses the variable’s label as its column header. If no label exists, the definition uses the variable’s name as the column header.

Featured in: All examples

overflow-control

determines the PUT statement behavior when the output pointer attempts to move past the last ODS column in the buffer. *overflow-control* is one of the following:

DROPOVER

discards items when a PUT statement attempts to write to ODS columns 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 to an ODS column 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 to an ODS column 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

PRINT

is a reserved fileref that directs the output that is produced by any PUT statements to all open ODS destinations.

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

Featured in: Example 1 on page 114

ODS Suboptions

To do this ...	Use this suboption
Specify one or more columns for the data component	COLUMNS= or VARIABLES=
Specify default values for column attributes that exist in the table definition, but that get their values from the data component	DYNAMIC=
Assert that all column definitions in the table definition can or cannot be used by more than one variable	GENERIC=
Specify a column header to use for any column that does not have a column header specified in the COLUMNS= or VARIABLES= suboption	LABEL=
Specify a name for the output object that the DATA step produces	OBJECT=
Specify a label for the output object that the DATA step produces	OBJECTLABEL=
Specify the table definition to use with the data component to produce the output object	TEMPLATE=

COLUMNS=(*column-specification(s)*)

specifies one or more columns for the data component. Each *column-specification* associates a DATA step variable with a column that is defined in the table definition.

The order of the columns in the data component is determined by their order in the COLUMNS= suboption.

Note: By default, 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. You can override this 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. (See the discussion of ORDER_DATA= on page 216.) Δ

Each *column-specification* has this general form:

column-name<=*variable-name*><(*attribute(s)*)>

column-name

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

Restriction: *column-name* must conform to the rules for SAS variable names. For information on these rules, see “Rules for Words and Names” in *SAS Language Reference: Dictionary*.

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

Featured in: Example 4 on page 126

variable-name

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

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

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

Featured in: Example 4 on page 126

attribute

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 the DATA step sets. You assign attributes with the following 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. For details, see the discussion of DYNAMIC= on page 109.

Featured in: Example 4 on page 126

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 on page 126

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

GENERIC=ON | OFF

specifies whether or not the DATA step uses this column definition for multiple variables. For details, see **GENERIC=** on page 109.

Default: OFF

Featured in: Example 4 on page 126

LABEL='column-label'

specifies a label for this particular column. For details, see **LABEL=** on page 110.

Featured in: Example 3 on page 121

Default: If you do not specify **COLUMNS=** or **VARIABLES=**, 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 **COLUMNS=** suboption in a FILE statement.

Interaction: 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 statement.

DYNAMIC=(dynamic-specification(s))

specifies default values for dynamic attribute values. Columns that do not contain their own **DYNAMIC=** specifications use these.

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

Each *dynamic-specification* has the following form:

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

dynamic-value-name

is the name that the table definition gives to a dynamic attribute value (see "DYNAMIC Statement" on page 222).

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.

Tip: By default, **DYNAMIC=** applies to all columns in the data component. You can override this specification for an individual column by specifying **DYNAMIC=** as an attribute for that column in the **COLUMNS=** or the **VARIABLES=** suboption.

GENERIC=ON | OFF

asserts that the DATA step does or does not use all column definitions for multiple variables.

ON

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

OFF

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

Default: OFF

Interaction: If you do not specify a table definition (see `TEMPLATE=` on page 110), `GENERIC=` is set to ON.

Restriction: Unless `GENERIC=ON` is specified in the `COLUMNS=` option, in the `ODS=` option, and in the table definition that you are using, ODS does not recognize the column names as a match.

Tip: By default, `GENERIC=` applies to all columns in the data component. You can override this specification for an individual column by specifying `GENERIC=` 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: ODS uses for the column header the first of these labels that it finds:

- a label that is specified with `HEADER=` for a particular column in the table definition (see the discussion of the column attribute `HEADER=` on page 165).
- a label that is specified for a particular column with `LABEL=` in the `COLUMNS=` or `VARIABLES=` suboption
- a label that is specified with `LABEL=` in the `ODS=` option
- a label that is assigned with the `LABEL` statement in the `DATA` step.

If no label is specified, the contents of the table definition determines whether the column header contains the variable name or is blank.

Featured in: Example 4 on page 126

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 these items that ODS finds:

- the object's label (see `OBJECTLABEL=` on page 110)
- 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 on these rules, see "Rules for Words and Names" in *SAS Language Reference: Dictionary*.

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 these 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 the discussion of `OBJECT=` on page 110)
- 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.

Featured in: Example 3 on page 121

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. By default, ODS first looks for *table-definition-name* in SASUSER.TEMPLAT. If it doesn't find the definition there, it looks in SASHELP.TMPLMST. You can change the locations that it searches with the ODS PATH statement (see "ODS PATH Statement" on page 64).

Default: base.datastep.table

Interaction: When you use the default table definition, GENERIC= is set to ON for all columns in the data component. (See GENERIC= on page 109.)

Featured in: Example 4 on page 126

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.

Note: By default, 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. You can override this 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. (See the discussion of the ORDER_DATA= on page 216.) △

Each *variable-specification* has this general form:

variable-name<=*column-name*><(*attribute(s)*)>

variable-name

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

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

Featured in: Example 2 on page 117 and Example 3 on page 121

column-name

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

Default: If you omit *column-name*, ODS looks for a column in the table definition that is named *variable-name* and places the variable in that column. If no such column exists, 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 on these rules, see "Rules for Words and Names" in *SAS Language Reference: Dictionary*.

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

attribute

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. You assign attributes just as you do in the COLUMNS= suboption (see the discussion of attributes on page 108).

Default: If you do not specify COLUMNS= or VARIABLES=, 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 statement.

Interaction: 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 statement.

Tip: `VARIABLES=` 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 don't need to specify a column name.

Featured in: Example 2 on page 117 and Example 3 on page 121

`_ODS_` in the PUT Statement

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

Arguments

specification

specifies the variables to write and where to write them. Each specification has the following form:

```
<ods-pointer-control>variable
```

variable

identifies the variable to write.

Featured in: Example 4 on page 126

ods-pointer-control

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

See also: “Column Pointer Controls and ODS” on page 113 and “Line Pointer Controls” on page 114

Options

`_ODS_`

moves data values for all columns to a buffer. The order of these columns is determined by the order specified by the `COLUMNS=` or `VARIABLES=` suboption in the ODS option in the FILE statement. If you do not specify either of these options, the order of the variables in the program data vector determines their order in the buffer.

The PUT statement writes this buffer to the data component.

Restriction: Use `_ODS_` only if you have previously specified the ODS option in the FILE statement. (See “FILE Statement for ODS” on page 106.)

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, but it may affect performance.

Tip: The order of the columns in the data component matches the order of the columns in buffer. However, the table definition that is combined with the data component to produce the output object may override this order. (See the discussion of the `ORDER_DATA=` on page 216 table attribute.)

`@` | `@@`

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 @@, each PUT statement in a DATA step writes a new line to the buffer.

Column Pointer Controls and ODS

Column pointer controls in a DATA step that uses ODS differ slightly from column pointer controls in a DATA step that does not use ODS. ODS is not character-based. Therefore, in ODS a column contains the entire value of a variable. Column 1 contains the first variable in the output; column 2 contains the second variable, and so on.

Column pointer controls have the following general forms:

@ods-column

+ods-column

@'column-name'

@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.

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

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

If the variable or expression is ...	SAS does this
not an integer	truncates the decimal portion and uses only the integer value
0 or negative	moves the pointer to column 1

Tip: By default, if *ods-column* exceeds the number of columns in the data component, ODS writes the current line, moves the pointer to the first column on the next line, and continues to process the PUT statement. You can alter this behavior with options in the FILE statement. (See the discussion of *overflow-control* on page 107.)

Featured in: Example 4 on page 126

+ods-column

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

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

Tip: If *ods-column* is greater than 0, the pointer moves to the right. If *ods-column* is less than 0, the pointer moves to the left. If *ods-column* is equal to 0, the pointer does not move.

If the current column position becomes less than 1, the pointer moves to column 1. If the current column position exceeds the number of columns in the data component, ODS writes the current line, moves the pointer to the first column on the next line, and continues to process the PUT statement.

See also: “When the Pointer Goes Past the End of a Line” on page 114

@'column-name'

moves the pointer to the ODS column identified by *'column-name'*.

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. Line pointer controls have the following general forms:

#line

/

#line

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

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

/

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

Featured in: Example 4 on page 126

When the Pointer Goes Past the End of a Line

In a DATA step that uses the Output Delivery System, the number of columns that is specified by the COLUMNS= or VARIABLES= suboption to the ODS option in the FILE statement determines the number of columns in the buffer and, eventually, in the data component. If you do not specify either of these options, the number of variables in the program data vector determines the number of columns.

Note: The table definition that is combined with the data component to produce the output object may change the number of columns that actually appear in the output object. \triangle

Using pointer controls and trailing @ or double trailing @, you may inadvertently position the pointer beyond the last column. You control how SAS handles this situation with options in the FILE statement. (See the discussion of *overflow-control on page 107*.)

Examples

Example 1: Using the Default Table Definition

ODS features:

FILE statement:

ODS option without suboptions

PRINT fileref

PUT _ODS_

This example uses the DATA step's default table definition to write an output object to the Listing destination.

Program

The OPTIONS statement controls several aspects of the Listing output.

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

The TITLE statement specifies a title for the output object.

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

PROC FORMAT creates a format for the variable Country.

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

This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object. The FORMAT statement assigns a format to the variable Country. The LABEL statement assigns a label to the variable Type.

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

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. Because no suboptions are specified, ODS uses the default DATA step table definition.

```
file print ods;
```

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 (see Output 4.1 on page 116).

```
put _ods_;
```

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

```
datalines;
1995 BRZ Wheat 1516
1995 BRZ Rice 11236
1995 BRZ Corn 36276
1995 CHN Wheat 102207
1995 CHN Rice 185226
1995 CHN Corn 112331
1995 IND Wheat 63007
1995 IND Rice 122372
1995 IND Corn 9800
```

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

Listing Output

Output 4.1 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.

Country	Leading Grain	Grain Producers Year	Kilotons	1
Brazil	Wheat	1995	1516	
Brazil	Rice	1995	11236	
Brazil	Corn	1995	36276	
China	Wheat	1995	102207	
China	Rice	1995	185226	
China	Corn	1995	112331	
India	Wheat	1995	63007	
India	Rice	1995	122372	
India	Corn	1995	9800	
Indonesia	Wheat	1995	.	
Indonesia	Rice	1995	49860	
Indonesia	Corn	1995	8223	
United States	Wheat	1995	59494	
United States	Rice	1995	7888	
United States	Corn	1995	187300	
Brazil	Wheat	1996	3302	
Brazil	Rice	1996	10035	
Brazil	Corn	1996	31975	
China	Wheat	1996	109000	
China	Rice	1996	190100	
China	Corn	1996	119350	
India	Wheat	1996	62620	
India	Rice	1996	120012	
India	Corn	1996	8660	
Indonesia	Wheat	1996	.	
Indonesia	Rice	1996	51165	
Indonesia	Corn	1996	8925	
United States	Wheat	1996	62099	
United States	Rice	1996	7771	
United States	Corn	1996	236064	

Example 2: Selecting Variables for the Data Component

ODS features:

FILE statement, ODS option:

VARIABLES= suboption

ODS HTML statement:

BODY=

URL= suboption

PUT _ODS_

Format: \$CNTRY. on page 115

This example selects variables to write to the data component. The output is routed to two ODS destinations: the Listing destination, which is open by default, and the HTML destination, which is opened by the ODS HTML statement.

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 1, "Alternative ODS HTML Statements for Running Examples in Different Operating Environments," on page 275. Δ

Program

The `OPTIONS` statement controls several aspects of the Listing output. None of these options affects the HTML output.

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

The `ODS HTML` statement opens the HTML destination and creates HTML output. It sends all output objects to the external file `selectvars-body.htm` in the current directory. Some browsers require an extension of `HTM` or `HTML` on the filename.

```
ods html body='selectvars-body.htm';
```

The `TITLE` statements provide titles for the data component.

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

This `DATA` step does not create a data set. Instead it creates a data component and, eventually, an output object. The `IF` statement restricts processing to observations with `Year=1996`. The `FORMAT` statement assigns a format to the variable `Country`. The `LABEL` statement assigns a label to the variable `Type`.

```
data _null_;
  length Country $ 3 Type $ 5;
  input Year country $ type $ Kilotons;
  if year=1996;
  format country $cntry.;
  label type='Grain';
```

The combination of the fileref `PRINT` and the `ODS` option in the `FILE` statement routes 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 the variables to write to the data component and the order in which to write them.

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

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 (see Display 4.1 on page 120 and Output 4.2 on page 121).

```
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. For a complete listing of the data, see the `DATALINES` statement on page 115.

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

... more lines of data ...
```

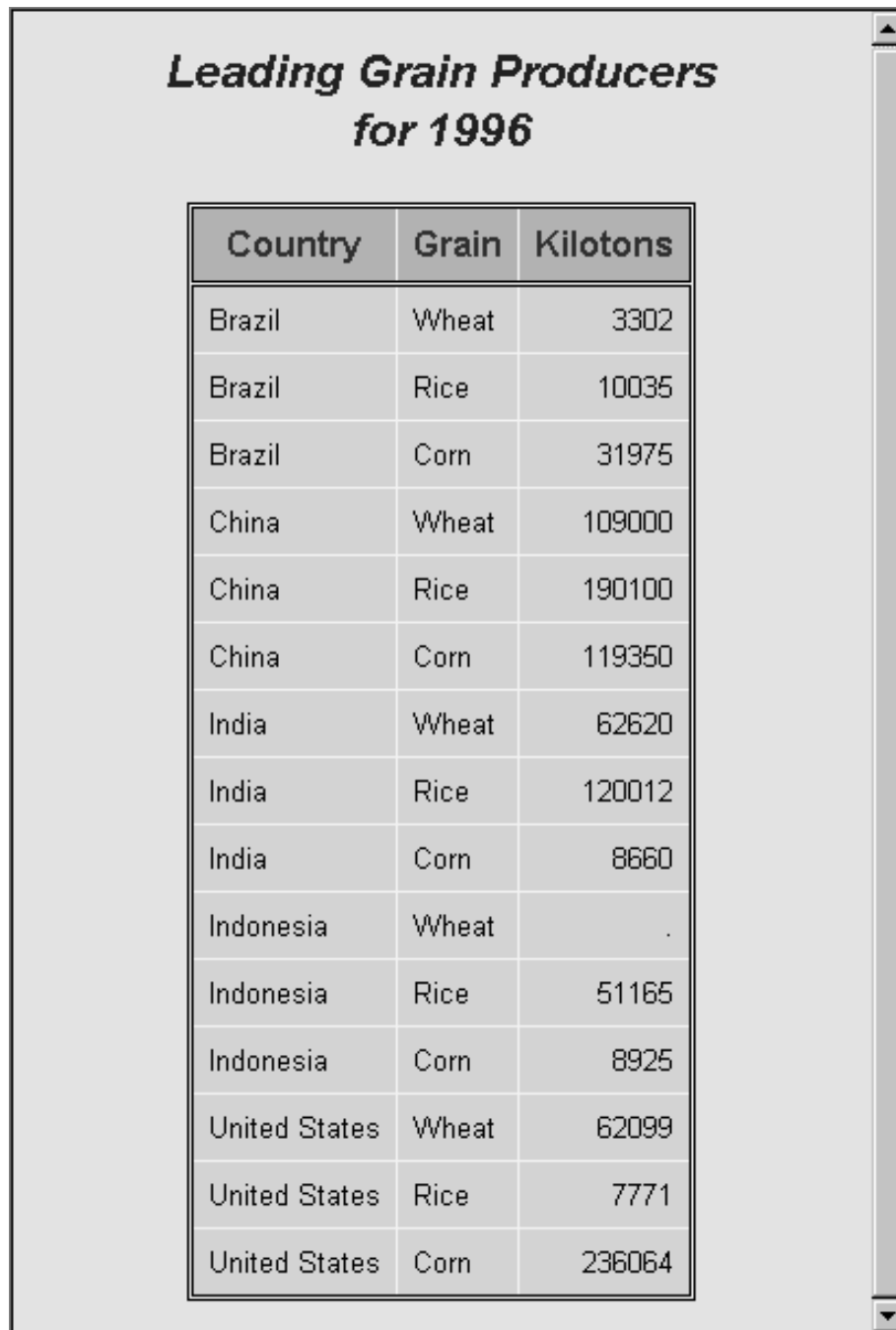
```
1996 USA Wheat 62099
1996 USA Rice 7771
1996 USA Corn 236064
;
```

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;
```

HTML Output

Display 4.1 Body File Produced by the HTML Destination



**Leading Grain Producers
for 1996**

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

Listing Output

Output 4.2 Listing Output Produced by the Listing Destination

Leading Grain Producers for 1996			1
Country	Grain	Kilotons	
Brazil	Wheat	3302	
Brazil	Rice	10035	
Brazil	Corn	31975	
China	Wheat	109000	
China	Rice	190100	
China	Corn	119350	
India	Wheat	62620	
India	Rice	120012	
India	Corn	8660	
Indonesia	Wheat	.	
Indonesia	Rice	51165	
Indonesia	Corn	8925	
United States	Wheat	62099	
United States	Rice	7771	
United States	Corn	236064	

Example 3: Specifying Attributes for a Column

ODS features:

FILE statement, ODS option:

OBJECTLABEL= suboption

VARIABLES= suboption

LABEL= attribute

FORMAT= attribute

PUT _ODS_

Format: \$CNTRY. on page 115

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 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 1, “Alternative ODS HTML Statements for Running Examples in Different Operating Environments,” on page 275. △

Program

The OPTIONS statement controls several aspects of the Listing output. NODATE and PAGENO= also affect the Printer output. None of these options affects the HTML output.

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

The ODS HTML statement opens the HTML destination and creates HTML output. Subsequent output objects go to the body file. CONTENTS= and FRAME= create a frame file that includes a table of contents that links to the contents of the body file. The body file also appears in the frame. The ODS PRINTER statement opens the Printer destination and creates Printer output. It sends all output objects to the external file `attrs.ps` in the current directory.

```
ods html body='attrs-body.htm'
         contents='attrs-contents.htm'
         frame='attrs-frame.htm';
ods printer file='attrs.ps';
```

The TITLE statements provide titles for the data component.

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

This DATA step does not create a data set. Instead it creates a data component and, eventually, an output object. The IF statement restricts processing to observations with Year=1996. The FORMAT statement assigns a format to the variable Country. The LABEL statement assigns a label to the variable Type.

```
data _null_;
  length Country $ 3 Type $ 5;
  input Year country $ type $ Kilotons;
  if year=1996;
  format country $cntry.;
  label type='Grain';
```

The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. Three ODS destinations— the Listing, the HTML, and the Printer destinations— are open. Because no table definition is specified, ODS uses the default DATA step definition.

```
file print ods=
```

OBJECTLABEL= specifies a label for the output object. This label appears in the Results folder and in the HTML contents file (see Display 4.2 on page 124).

```
(objectlabel='1996 Grain Production')
```

The VARIABLES= suboption specifies the variables to write to the data component and the order in which to write them. The suboption includes attribute specifications of a label (for the variable Type) and a format (for the variable Kilotons). 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.

```
variables=(country
           type(label='Type of Grain')
           kilotons(format=comma12.))
);
```

The _ODS_ option in the PUT statement writes to the buffer every variable that the PUT statement writes to the data component. It uses default attributes for Country, and it uses any attributes specified in the VARIABLES= suboption for the other variables. For attributes that are not specified in VARIABLES=, it uses the defaults.

```
put _ods_;
```

The data provide information on the amounts of wheat, rice, and corn that five leading grain-producing nations produced during 1995 and 1996. For a complete listing of the data, see the DATALINES statement on page 115.

```
datalines;  
1995 BRZ  Wheat    1516  
1995 BRZ  Rice     11236  
1995 BRZ  Corn     36276  
  
... more lines of data ...  
  
1996 USA  Wheat    62099  
1996 USA  Rice     7771  
1996 USA  Corn     236064  
;
```

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 PRINTER statement closes the Printer destination. You must close the destination before you can print the output on a physical printer.

```
ods html close;  
ods printer close;
```

HTML Output

Display 4.2 Frame File Produced by the HTML Destination

In this HTML frame file, the object's label, which was supplied by OBJECTLABEL=, appears in the table of contents as the link to the output object. In the body file, the label that is supplied as an attribute for the variable Grain becomes its column header. The format for Kilotons was also supplied as an attribute in the ODS= option in the FILE statement.

Table of Contents		<i>Leading Grain Producers for 1996</i>		
I. The Dastep Procedure -1996 Grain Production		Country	Type of Grain	Kilotons
		Brazil	Wheat	3,302
		Brazil	Rice	10,035
		Brazil	Corn	31,975
		China	Wheat	109,000
		China	Rice	190,100
		China	Corn	119,350
		India	Wheat	62,620
		India	Rice	120,012
		India	Corn	8,660
		Indonesia	Wheat	.
		Indonesia	Rice	51,165
		Indonesia	Corn	8,925
		United States	Wheat	62,099
		United States	Rice	7,771
		United States	Corn	236,064

Printer Output

Just as in the HTML body file and in the Listing output, in the Printer output the label that is supplied as an attribute for the variable Grain becomes its column header. The format for Kilotons was also supplied as an attribute in the ODS= option in the FILE statement.

Leading Grain Producers for 1996

1

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

Listing Output

Just as in the HTML body file and the Printer output, in the Listing output the label that is supplied as an attribute for the variable Grain becomes its column header. The format for Kilotons was also supplied as an attribute in the ODS= option in the FILE statement.

Leading Grain Producers for 1996			1
Country	Type of Grain	Kilotons	
Brazil	Wheat	3,302	
Brazil	Rice	10,035	
Brazil	Corn	31,975	
China	Wheat	109,000	
China	Rice	190,100	
China	Corn	119,350	
India	Wheat	62,620	
India	Rice	120,012	
India	Corn	8,660	
Indonesia	Wheat	.	
Indonesia	Rice	51,165	
Indonesia	Corn	8,925	
United States	Wheat	62,099	
United States	Rice	7,771	
United States	Corn	236,064	

Example 4: Using a Simple User-Defined Definition

ODS features:

PROC TEMPLATE

FILE statement, ODS option:

COLUMNS= suboption

FORMAT= attribute

DYNAMIC= attribute

GENERIC= attribute

TEMPLATE=

PUT _ODS_:

column pointer controls

line pointer controls

This example illustrates how to use a simple user-defined table definition in the DATA step. It also illustrates the use of pointer controls in the PUT _ODS_ statement.

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 1, "Alternative ODS HTML Statements for Running Examples in Different Operating Environments," on page 275. Δ

Program

This PROC TEMPLATE step defines a table definition named **phonelist**. The definition defines two columns: NAME and PHONE. The GENERIC=ON suboption 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 the definition. Thus, each variable can have a different column header. STYLE= specifies a style element, Data, to use as the basis for rendering the data in this column. The font face and font size that Data normally uses are replaced by the ones that are specified in STYLE=.

The header for PHONE is hardcoded as Telephone. STYLE= specifies a style element to use for the data in this column. For information on PROC TEMPLATE, see Chapter 5, “The TEMPLATE Procedure,” on page 131.

```
/* Define the table definition 'phonelist' */
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;
```

The ODS LISTING CLOSE statement closes the Listing destination to conserve resources. The Listing destination is open by default.

```
ods listing close;
```

The ODS HTML statement opens the HTML destination and creates HTML output. Subsequent output objects go to the body file.

```
ods html body='ptcntrl-body.htm';
```

The TITLE statement provides a title for the data component.

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

PROC FORMAT creates a format for telephone numbers.

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

The data set PHONES contains names and phone numbers for each person.

```

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

```

PROC SORT sorts the data set PHONES by LAST_NAME and replaces the original data set with the sorted one.

```

proc sort data=phones;
by last_name;
run;

```

This DATA step does not create a data set. Instead, it creates a data component and, eventually, an output object.

```

data _null_;
set phones;

```

The combination of the fileref PRINT and the ODS option in the FILE statement routes the results of the DATA step to ODS. Only the HTML destination is open. TEMPLATE= tells ODS to use the table definition named **phonelist**, which was just created with PROC TEMPLATE.

```

file print ods=(template='phonelist'

```

The COLUMNS= suboption places DATA step variables into columns that are defined in the table definition. Both LAST_NAME and FIRST_NAME go in columns that are defined as NAME in the table definition. GENERIC=ON must be set in both the table definition and the ODS= option in order for you to use a column definition for more than one column. The variable BUSINESS_PHONE is placed in a column that is defined as PHONE in the definition. The DYNAMIC= attribute assigns a value to the variable COLHEADER. This value is passed to the table definition when the output object is created, and the definition uses it for the column header. Thus, even though the variables use the same column definition from the table definition, the columns in the output object have different column headers. The FORMAT= attribute assigns the format PHONENUM. to the column named PHONE.

```

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

```


This piece of conditional code executes a different PUT _ODS_ statement for each of three conditions. If BUSINESS_PHONE is missing, 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. It then writes the value for HOME_PHONE in column 3, overwriting the missing value of BUSINESS_PHONE.

If HOME_PHONE is missing, the PUT statement simply writes values for LAST_NAME, FIRST_NAME, and BUSINESS_PHONE to the buffer.

Finally, if both phone numbers have values, the PUT statement writes values for LAST_NAME, FIRST_NAME, and BUSINESS_PHONE to the buffer in the first line. It then goes to the next line (as directed by the line pointer control /) and writes the value of HOME_PHONE in the third column of the next line.

```

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

```

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;
```

HTML Output*New Subscriber Telephone List*

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

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *The Complete Guide to the SAS® Output Delivery System, Version 8*, Cary, NC: SAS Institute Inc., 1999. 310 pp.

The Complete Guide to the SAS® Output Delivery System, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-425-X

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st printing, October 1999

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

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

The Institute is a private company devoted to the support and further development of its software and related services.