Base SAS® 9.4 Utilities: Reference

SAS® Documentation
July 24, 2020

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *Base SAS® 9.4 Utilities: Reference*. Cary, NC: SAS Institute Inc.

Base SAS® 9.4 Utilities: Reference

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

August 2020

9.4-P9:lebaseutilref

Contents

Syntax Conventions for the SAS Language

Overview of Syntax Conventions for the SAS Language

SAS uses standard conventions in the documentation of syntax for SAS language elements. These conventions enable you to easily identify the components of SAS syntax. The conventions can be divided into these parts:

- syntax components
- style conventions
- special characters
- references to SAS libraries and external files

Syntax Components

The components of the syntax for most language elements include a keyword and arguments. For some language elements, only a keyword is necessary. For other language elements, the keyword is followed by an equal sign (=). The syntax for arguments has multiple forms in order to demonstrate the syntax of multiple arguments, with and without punctuation.

keyword
specifies the name of the SAS language element that you use when you write your program. Keyword is a literal that is usually the first word in the syntax. In a CALL routine, the first two words are keywords.

In these examples of SAS syntax, the keywords are bold:

CHAR (*string, position*)

CALL RANBIN (*seed, n, p, x*);

ALTER (*alter-password*)

BEST *w*.

REMOVE <*data-set-name*>

In this example, the first two words of the CALL routine are the keywords:

CALL RANBIN(*seed*, *n*, *p*, *x*)

The syntax of some SAS statements consists of a single keyword without arguments:

DO;
... *SAS code* ...
END;

Some system options require that one of two keyword values be specified:

DUPLEX | **NODUPLEX**

Some procedure statements have multiple keywords throughout the statement syntax:

CREATE <UNIQUE> **INDEX** *index-name* **ON** *table-name* (*column-1* <, *column-2*, …>)

argument
specifies a numeric or character constant, variable, or expression. Arguments follow the keyword or an equal sign after the keyword. The arguments are used by SAS to process the language element. Arguments can be required or optional. In the syntax, optional arguments are enclosed in angle brackets (< >).

In this example, *string* and *position* follow the keyword CHAR. These arguments are required arguments for the CHAR function:

CHAR (*string, position*)

Each argument has a value. In this example of SAS code, the argument *string* has a value of 'summer', and the argument *position* has a value of 4:

```
x=char('summer', 4);
```

In this example, *string* and *substring* are required arguments, whereas *modifiers* and *startpos* are optional.

FIND(*string, substring* <, *modifiers*> <, *startpos*>

argument(s)
specifies that one argument is required and that multiple arguments are allowed. Separate arguments with a space. Punctuation, such as a comma (,) is not required between arguments.

The MISSING statement is an example of this form of multiple arguments:

MISSING *character(s)*;

<LITERAL_ARGUMENT> *argument-1* <<LITERAL_ARGUMENT> *argument-2* ... >
specifies that one argument is required and that a literal argument can be associated with the argument. You can specify multiple literals and argument pairs. No punctuation is required between the literal and argument pairs. The ellipsis (...) indicates that additional literals and arguments are allowed.

The BY statement is an example of this argument:

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>;

argument-1 <*options*> <*argument-2* <*options*> ...>
specifies that one argument is required and that one or more options can be associated with the argument. You can specify multiple arguments and associated options. No punctuation is required between the argument and the

option. The ellipsis (...) indicates that additional arguments with an associated option are allowed.

The FORMAT procedure PICTURE statement is an example of this form of multiple arguments:

PICTURE name <(*format-options*)>
<*value-range-set-1* <(*picture-1-options*)>
<*value-range-set-2* <(*picture-2-options*)> …>>;

argument-1=value-1 <argument-2=value-2 ...>
specifies that the argument must be assigned a value and that you can specify multiple arguments. The ellipsis (...) indicates that additional arguments are allowed. No punctuation is required between arguments.

The LABEL statement is an example of this form of multiple arguments:

LABEL *variable-1=label-1 <variable-2=label-2* …>;

argument-1 <, *argument-2*, ...>
specifies that one argument is required and that you can specify multiple arguments that are separated by a comma or other punctuation. The ellipsis (...) indicates a continuation of the arguments, separated by a comma. Both forms are used in the SAS documentation.

Here are examples of this form of multiple arguments:

AUTHPROVIDERDOMAIN (*provider-1:domain-1* <, *provider-2:domain-2*, …>

INTO :*macro-variable-specification-1* <, :*macro-variable-specification-2*, …>

Note: In most cases, example code in SAS documentation is written in lowercase with a monospace font. You can use uppercase, lowercase, or mixed case in the code that you write.

Style Conventions

The style conventions that are used in documenting SAS syntax include uppercase bold, uppercase, and italic:

UPPERCASE BOLD
identifies SAS keywords such as the names of functions or statements. In this example, the keyword ERROR is written in uppercase bold:

ERROR <*message*>;

UPPERCASE
identifies arguments that are literals.

In this example of the CMPMODEL= system option, the literals include BOTH, CATALOG, and XML:

CMPMODEL=BOTH | CATALOG | XML |

italic
identifies arguments or values that you supply. Items in italic represent user-supplied values that are either one of the following:

- nonliteral arguments. In this example of the LINK statement, the argument *label* is a user-supplied value and therefore appears in italic:

  LINK *label*;

- nonliteral values that are assigned to an argument.

  In this example of the FORMAT statement, the argument DEFAULT is assigned the variable *default-format*:

  FORMAT *variable(s)* <*format* > <DEFAULT = *default-format*>;

Special Characters

The syntax of SAS language elements can contain the following special characters:

=
an equal sign identifies a value for a literal in some language elements such as system options.

In this example of the MAPS system option, the equal sign sets the value of MAPS:

MAPS=*location-of-maps*

< >
angle brackets identify optional arguments. A required argument is not enclosed in angle brackets.

In this example of the CAT function, at least one item is required:

CAT (*item-1* <, *item-2*, …>)

|
a vertical bar indicates that you can choose one value from a group of values. Values that are separated by the vertical bar are mutually exclusive.

In this example of the CMPMODEL= system option, you can choose only one of the arguments:

CMPMODEL=BOTH | CATALOG | XML

...
an ellipsis indicates that the argument can be repeated. If an argument and the ellipsis are enclosed in angle brackets, then the argument is optional. The repeated argument must contain punctuation if it appears before or after the argument.

In this example of the CAT function, multiple *item* arguments are allowed, and they must be separated by a comma:

CAT (*item-1 <, item-2*, …>)

'*value*' or "*value*"
indicates that an argument that is enclosed in single or double quotation marks must have a value that is also enclosed in single or double quotation marks.

In this example of the FOOTNOTE statement, the argument *text* is enclosed in quotation marks:

FOOTNOTE <*n*> <*ods-format-options* '*text*' | "*text*">;

;

a semicolon indicates the end of a statement or CALL routine.

In this example, each statement ends with a semicolon:

```
data namegame;
   length color name $8;
   color = 'black';
   name = 'jack';
   game = trim(color) || name;
run;
```

References to SAS Libraries and External Files

Many SAS statements and other language elements refer to SAS libraries and external files. You can choose whether to make the reference through a logical name (a libref or fileref) or use the physical filename enclosed in quotation marks.

If you use a logical name, you typically have a choice of using a SAS statement (LIBNAME or FILENAME) or the operating environment's control language to make the reference. Several methods of referring to SAS libraries and external files are available, and some of these methods depend on your operating environment.

In the examples that use external files, SAS documentation uses the italicized phrase *file-specification*. In the examples that use SAS libraries, SAS documentation uses the italicized phrase *SAS-library* enclosed in quotation marks:

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

1

Dictionary of Utilities

Dictionary

%DS2CSV Macro

Converts SAS data sets to comma-separated value (CSV) files.

Restriction: This macro cannot be used in a DATA step. Run the macro only in open code.

Syntax

%DS2CSV(*argument-1=value-1*, *argument-2=value-2* <,*argument-3=value-3* …>)

Arguments That Affect Input and Output

csvfile=*external-filename*
specifies the name of the CSV file where the formatted output is to be written. If the file that you specify does not exist, then it is created for you.

Note Do not use the CSVFILE argument if you use the CSVFREF argument.

csvfref=*fileref*
specifies the SAS fileref that points to the location of the CSV file where the formatted output is to be written. If the file that you specify does not exist, then it is created for you.

Note Do not use the CSVFREF argument if you use the CSVFILE argument.

openmode=REPLACE | APPEND
indicates whether the new CSV output overwrites the information that is currently in the specified file or if the new output is appended to the end of the existing file. The default value is REPLACE. If you do not want to replace the current contents, then specify OPENMODE=APPEND to add your new CSV-formatted output to the end of an existing file.

Note OPENMODE=APPEND is not valid if you are writing your resulting output to a partitioned data set (PDS) on z/OS.

Arguments That Affect MIME and HTTP Headers

For more information about MIME and HTTP headers, see the Internet Request for Comments (RFC) documents RFC 1521 and RFC 1945, respectively.

conttype=Y | N
indicates whether to write a content type header. This header is written by default.

Restriction This argument is valid only when RUNMODE=S.

contdisp=Y | N
indicates whether to write a content disposition header. This header is written by default.

Restriction This argument is valid only when RUNMODE=S.

Note If you specify CONTDISP=N, then the SAVEFILE argument is ignored.

mimehdr1=*MIME/HTTP-header*
specifies the text that is to be used for the first MIME or HTTP header that is written. This header is written after the content type and disposition headers. By default, nothing is written for this header.

Restriction This argument is valid only when RUNMODE=S.

mimehdr2=*MIME/HTTP-header*
specifies the text that is to be used for the second MIME or HTTP header that is written. This header is written after the content type and disposition headers. By default, nothing is written for this header.

Restriction This argument is valid only when RUNMODE=S.

mimehdr3=*MIME/HTTP-header*
specifies the text that is to be used for the third MIME or HTTP header that is written when RUNMODE=S is specified. This header is written after the content type and disposition headers. By default, nothing is written for this header.

Restriction This argument is valid only when RUNMODE=S.

mimehdr4=*MIME/HTTP-header*
specifies the text that is to be used for the fourth MIME or HTTP header that is written. This header is written after the content type and disposition headers. By default, nothing is written for this header.

Restriction This argument is valid only when RUNMODE=S.

mimehdr5=*MIME/HTTP-header*
specifies the text that is to be used for the fifth MIME or HTTP header that is written. This header is written after the content type and disposition headers. By default, nothing is written for this header.

runmode=S | B
specifies whether you are running the %DS2CSV macro in batch or server mode. The default setting for this argument is RUNMODE=S.

- *Server mode* (RUNMODE=S) is used with Application Dispatcher programs and streaming output stored processes. Server mode causes DS2CSV to generate appropriate MIME or HTTP headers. For more information, see SAS/IntrNet Software.
- *Batch mode* (RUNMODE=B) means that you are submitting the DS2CSV macro in the SAS Program Editor or that you included it in a SAS program.

Note: No HTTP headers are written when you specify batch mode.

Restriction RUNMODE=S is valid only when used within the SAS/IntrNet and Stored Process servers.

savefile=*filename*
specifies the filename to display in the Web browser's **Save As** dialog box. The default value is the name of the data set plus ".csv".

Restriction This argument is valid only when RUNMODE=S.

Note This argument is ignored if CONTDISP=N is specified.

Arguments That Affect CSV Creation

colhead=Y | N
indicates whether to include column headings in the CSV file. The column headings that are used depend on the setting of the LABELS argument. By default, column headings are included as the first record of the CSV file.

data=*SAS-data-set-name*
specifies the SAS data set that contains the data that you want to convert into a CSV file. This argument is required. However, if you omit the data set name, DS2CSV attempts to use the most recently created SAS data set.

formats=Y | N
indicates whether to apply the data set's defined variable formats to the values in the CSV file. By default, all formats are applied to values before they are added to the CSV file. The formats must be stored in the data set in order for them to be applied.

labels=Y | N
indicates whether to use the SAS variable labels that are defined in the data set as your column headings. The DS2CSV macro uses the variable labels by default. If a variable does not have a SAS label, then use the name of the variable. Specify labels=N to use variable names instead of the SAS labels as your column headings.

See For more information about column headings, see the colhead argument.

pw=*password*
specifies the password that is needed to access a password-protected data set. This argument is required if the data set has a READ or PW password. (You do not need to specify this argument if the data set has only WRITE or ALTER passwords.)

sepchar=*separator-character*
specifies the character that is used for the separator character. Specify the two-character hexadecimal code for the character or omit this argument to get the default setting. The default settings are 2C for ASCII systems and 6B for EBCDIC systems. (These settings represent commas (,) on their respective systems.)

var=*variable(s)*
specifies one or more variables that are to be included in the CSV file and the order in which they should be included. To include all of the variables in the data set, do not specify this argument. Separate multiple variables with a single blank space. Do not use a comma in the list of variable names.

Restriction A range of values is not valid. For example, `var1-var4`.

where=*where-expression*
specifies a valid WHERE clause that selects observations from the SAS data set. Using this argument subsets your data based on the criteria that you supply for *where-expression*.

Details

The DS2CSV macro converts SAS data sets to comma-separated value (CSV) files. You can specify the hexadecimal code for the separator character if you want to create some other type of output file (for example, a tab-separated value file).

Example

The following example uses the %DS2CSV macro to convert the SASHELP.RETAIL data set to a comma-separated value file:

```
%ds2csv (data=sashelp.retail, runmode=b, csvfile=c:\temp\retail.csv);
```

%TSLIT Macro

Overrides the need for double quotation marks around literal text and puts single quotation marks around the input value.

Restriction: The macro facility does not run within Cloud Analytic Services (CAS). The macro facility runs in a SAS client session within SAS Viya.

Syntax

%TSLIT (*literal text*)

Details

To reference a macro variable in a delimited identifier, use the SAS macro %TSLIT. This SAS macro overrides the need for double quotation marks around the literal text and puts single quotation marks around the input value.

The %TSLIT macro is stored in the default autocall macro library. Here is an example using %TSLIT.

```
%let profit=%str($100,000);

data _null_;
    put %tslit(PROFIT: &profit);
run;
```

Here is the output to the SAS log:

```
PROFIT: $100,000
```

Here is an example outside a DATA step:

```
%put %tslit(&profit);
```

Here is the output to the SAS log:

```
'$100,000'
```

Here is an example using PROC FEDSQL:

```
proc fedsql;
    CREATE TABLE tab1(var1 CHAR(10));
    INSERT INTO tab1 VALUES(%tslit(&SYSHOSTNAME));

--- The following will produce an error because it ---
--- thinks it is looking for a column name.     ---

    INSERT INTO tab1 VALUES("&SYSHOSTNAME");
    SELECT * FROM tab1;
    DROP TABLE tab1;
quit;
```

Here is the output to the SAS log:

```
1    proc fedsql;
NOTE: Writing HTML Body file: sashtml.htm
2         CREATE TABLE tab1(var1 CHAR(10));
NOTE: Execution succeeded. No rows affected.
3         INSERT INTO tab1 VALUES(%tslit(&SYSHOSTNAME));
NOTE: Execution succeeded. One row affected.
4
5    --- The following will produce an error because it ---
6    --- thinks it is looking for a column name.     ---
7
8         INSERT INTO tab1 VALUES("SYSHOSTNAME");
ERROR: Syntax error or access violation
9         SELECT * FROM tab1;
10        DROP TABLE tab1;
NOTE: Execution succeeded. No rows affected.
11   quit;
```