APPENDIX
1

Rules for Words and Names

Words in the SAS Language

Definition

A *word* or *token* in the SAS language is a collection of characters that communicates a meaning to SAS and is not divisible into smaller units capable of independent use. It can contain a maximum of 32,767 characters.

A word or token ends when SAS encounters one of the following:

- the beginning of a new token
- a blank after a name or a number token
- the ending quotation mark of a literal token.

Each word or token in the SAS language belongs to one of four categories:

- names
- literals
- numbers
- special characters.

Types of Words or Tokens

There are four basic types of words or tokens:

name
is a series of characters that begin with a letter or an underscore. Later characters can include letters, underscores, and numeric digits. A name token can contain up to 32,767 characters. In most contexts, however, SAS names are limited to a shorter maximum length, such as 32 or 8 characters. See Table A1.1 on page 1185. Examples of name tokens include:

- `data`
- `_new`
- `yearcutoff`
- `year_99`
- `descending`
- `_n_`

literal
consists of 1 to 32,767 characters enclosed in single or double quotation marks. Examples of literals include

- `'Chicago'`
- `"1990-91"`
- `'Amelia Earhart'`
- `'Amelia Earhart''s plane'`
- `"Report for the Third Quarter"`

Note: The surrounding quotation marks identify the token as a literal, but SAS does not store these marks as part of the literal token. △

number
in general is composed entirely of numeric digits, with an optional decimal point and a leading plus or minus sign. SAS also recognizes numeric values in the folllowing forms as number tokens: scientific (E−) notation, hexadecimal notation, missing value symbols, and date and time literals. Examples of number tokens include

- `5683`
- `2.35`
- `0b0x`
- `-5`
- `5.4E-1`
- `'24aug90'd`

special character
is usually any single keyboard character other than letters, numbers, the underscore, and the blank. In general, each special character is a single token, although some two-character operators, such as ** and <=, form single tokens.

The blank can end a name or a number token, but it is not a token. Examples of special-character tokens include

- `=`
- `;`
- `'`
- `+`
- `@`
- `/`

Placement and Spacing of Words in SAS Statements

Spacing Requirements

1. You can begin SAS statements in any column of a line and write several statements on the same line.
2. You can begin a statement on one line and continue it on another line, but you cannot split a word between two lines.
3. A blank is not treated as a character in a SAS statement unless it is enclosed in quotation marks as a literal or part of a literal. Therefore, you can put multiple blanks any place in a SAS statement where you can put a single blank, with no effect on the syntax.
4. The rules for recognizing the boundaries of words or tokens determine the use of spacing between them in SAS programs. If SAS can determine the beginning of each token due to cues such as operators, you do not need to include blanks. If SAS cannot determine the beginning of each token, you must use blanks. See "Examples" on page 1183.

Although SAS does not have rigid spacing requirements, SAS programs are easier to read and maintain if you consistently indent statements. The examples in this book illustrate useful spacing conventions.

Examples

- In this statement, blanks are *not required* because SAS can determine the boundary of every token by examining the beginning of the next token:

  ```
  total=x+y;
  ```

  The first special-character token, the equal sign, marks the end of the name token **total**. The plus sign, another special-character token, marks the end of the name token **x**. The last special-character token, the semicolon, marks the end of the **y** token. Though blanks are not needed to end any tokens in this example, you may add them for readability, as shown here:

  ```
  total = x + y;
  ```

- This statement *requires blanks* because SAS cannot recognize the individual tokens without them:

```
input group 15 room 20;
```

Without blanks, the entire statement up to the semicolon fits the rules for a name token: it begins with a letter or underscore, contains letters, digits, or underscores thereafter, and is less than 32,767 characters long. Therefore, this statement requires blanks to distinguish individual name and number tokens.

Names in the SAS Language

Definition

A *SAS name* is a name token that represents

- □ variables
- □ SAS data sets
- □ formats or informats
- □ SAS procedures
- □ options
- □ arrays
- □ statement labels
- □ SAS macros or macro variables
- □ SAS catalog entries
- □ librefs or filerefs.

There are two kinds of names in SAS:

- □ names of elements of the SAS language
- □ names supplied by SAS users.

Rules for User-Supplied SAS Names

Rules for Most SAS Names

Note: The rules are more flexible for SAS variables names than for other language elements. See “Rules for SAS Variable Names” on page 1186. △

1 The length of a SAS name depends on the element it is assigned to. Many SAS names can be 32 characters long; others have a maximum length of 8. The notable exception is user-written informats, which have a maximum length of 7. See Table A1.1 on page 1185

.

2 The first character must be a letter (A, B, C, . . ., Z) or underscore (_). Subsequent characters can be letters, numeric digits (0, 1, . . ., 9), or underscores.

3 You can use upper or lowercase letters. SAS processes names as uppercase regardless of how you type them.

4 Blanks cannot appear in SAS names.

5 Special characters, except for the underscore, are not allowed. In filerefs only, you can use the dollar sign ($), pound sign (#), and at sign (@).

6 SAS reserves a few names for automatic variables and variable lists, SAS data sets, and librefs.

- **a** When creating variables, do not use the names of special SAS automatic variables (for example, _N_ and _ERROR_) or special variable list names (for example, _CHARACTER_, _NUMERIC_, and _ALL_).
- **b** When associating a libref with a SAS data library, do not use these:

  SASHELP

  SASMSG

  SASUSER

  WORK
- **c** When you create SAS data sets, do not use these names:

  NULL

  DATA

  LAST

7 When assigning a fileref to an external file, do not use:

SASCAT

8 When you create a macro variable, do not use names that begin with **`SYS`**.

Table A1.1 Maximum Length of User-Supplied SAS Names

SAS Language Element	Maximum Length
Members of SAS data libraries (SAS data sets, views, catalogs, indexes) except for generation data sets	32
Generation data sets	28
Catalog entries	32
Engines	8
Librefs	8
Filerefs	8
Passwords	8
DATA step variables	32
DATA step variable labels	256
DATA step statement labels	32
Arrays	32
DATA step windows	32
Functions	16
CALL routines	16
Formats	8
Informats	7
Macros	32

SAS Language Element	Maximum Length
Macro variables	32
Macro windows	32
SCL variables	32
SAS/EIS items	depends on the data dictionary
Procedure names (First 8 characters must be unique, and may not begin with "SAS".)	16

Rules for SAS Variable Names

The rules for SAS variable names have expanded to provide more functionality. The setting of the VALIDVARNAME= system option determines what rules apply to the variables that you can create and process in your SAS session as well as to variables that you want to read from existing data sets. The VALIDVARNAME= option has four settings (V7, UPCASE, V6 and ANY), each with varying degrees of flexibility for variable names:

V7
: is the default setting for Version 7.

  Variable name rules for Version 7 are as follows:

  1. SAS variable names may be up to 32 characters in length.
  2. The first character must begin with an alphabetic character or an underscore. Subsequent characters can be alphabetic characters, numeric digits, or underscores.
  3. A variable name may not contain blanks.
  4. A variable name may not contain any special characters other than the underscore.
  5. A variable name may contain mixed case. The mixed case is remembered and used for presentation purposes only. (SAS stores the case used in the first reference to a variable.) When SAS processes variable names, however, it internally uppercases them. You cannot, therefore, use the same letters with different combinations of lower- and uppercase to represent different variables. For example, **`cat`**, **`Cat`**, and **`CAT`** all represent the same variable.
  6. You may not assign the names of special SAS automatic variables (such as _N_ and _ERROR_) or variable list names (such as _NUMERIC_, _CHARACTER_, and _ALL_) to variables.

UPCASE
: is the same as when VALIDVARNAME=V7, except that variable names are uppercased, as in earlier versions of SAS software.

V6
: is the same as when VALIDVARNAME=V7, *except* that variable names are uppercased and they may be only 8 characters long, as in earlier versions of SAS software.

  CAUTION:

  Transitional use only: VALIDVARNAME=V6 is intended for transitional use only. Use this option for applications that you need to use under both Version 6 and Version 7 of the SAS System. △

CAUTION:

May render some applications unusable: Using VALIDVARNAME=V6 may render some Version 7 macros or Version 7 SAS/AF applications unusable. △

ANY

1 SAS variable names may be up to 32 characters in length.

2 The name may start with or contain any characters, including blanks.

Note: If you use any characters other the ones that are valid when VALIDVARNAME=V7 (letters, numeric digits, or underscores), then you must express the variable name as a *name literal* and you must set VALIDVARNAME=ANY. See “SAS Name Literals” on page 1187. △

3 A variable name may contain mixed case. The mixed case is remembered and used for presentation purposes only. (SAS stores the case used in the first reference to a variable.) When SAS processes variable names, however, it internally uppercases them. You cannot, therefore, use the same letters with different combinations of lower- and uppercase to represent different variables. For example, **`cat`**, **`Cat`**, and **`CAT`** all represent the same variable.

SAS Name Literals

Definition

A *SAS name literal* is a name token that is expressed as a string within quotation marks, followed by the letter *n*. Name literals allow you to use special characters (or blanks) that are not otherwise allowed in SAS names when you specify a SAS data set or a variable. Name literals are especially useful for expressing DBMS column and table names that contain special characters.

Important Restrictions

- □ You can use a name literal only for variables, statement labels, and DBMS column and table names.
- □ You can use name literals in a DATA step or a PROC SQL step only.
- □ When the name literal of a variable or DBMS column contains any characters that are not allowed when VALIDVARNAME=V7, then you must set the system option VALIDVARNAME=ANY.
- □ When the name literal of a DBMS table or column contains any characters not allowed by SAS rules for names, then you may need to specify a SAS/ACCESS LIBNAME statement option.

Note: For more details and examples about the SAS/ACCESS LIBNAME statement and about using DBMS table and column names that do not conform to SAS naming conventions, see *SAS/ACCESS Software for Relational Databases: Reference*. △

Examples

Examples of SAS name literals are

□
```
libname foo SAS/ACCESS-engine-name
            SAS/ACCESS-engine-connection-options;
data foo.'My Table'n;
```

```
□ input 'Amount Budgeted'n 'Amount Spent'n
        'Amount Difference'n;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Language Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS® Language Reference, Version 8

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

ISBN 1–58025–369–5

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

1st printing, October 1999