# Contents

*About This Book* ................................................................. v
*What’s New in SAS 9.4 Data Set Options* ................................. ix

## Chapter 1 • About Data Set Options ........................................ 1
- Definition of Data Set Options .......................................... 1
- Syntax ........................................................................... 1
- Using Data Set Options ................................................... 2
- Data Set Options Documented in Other SAS Publications .......... 3

## Chapter 2 • Dictionary of Data Set Options ................................ 5
- Data Set Options by Category ............................................ 6
- Dictionary ....................................................................... 8

*Recommended Reading* ......................................................... 79
*Index* ............................................................................. 81
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 (=).

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 the following examples of SAS syntax, the keywords are the first words in the syntax:

- **CHAR** *(string, position)*
- **CALL RANBIN** *(seed, n, p, x)*;
- **ALTER** *(alter-password)*
- **BEST** *w.*
- **REMOVE** <**data-set-name**>

In the following 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* ...
Some system options require that one of two keyword values be specified:

**DUPLEX | NODUPLEX**

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 between angle brackets.

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

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

In the following example, *string* and *substring* are required arguments, while
*modifiers* and *startpos* are optional.

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

*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 the
following example, the keyword **ERROR** is written in uppercase bold:

```sas
ERROR<message>;
```

**UPPERCASE**
identifies arguments that are literals.

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

```sas
CMPMODEL = BOTH | CATALOG | XML
```

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

- nonliteral arguments
  In the following example of the LINK statement, the argument *label* is a user-
supplied value and is therefore written in italics:

```sas
LINK label;
```

- nonliteral values that are assigned to an argument
  In the following example of the FORMAT statement, the argument DEFAULT is
assigned the variable *default-format*:
FORMAT = variable-1 <, ..., variable-n format >=<DEFAULT = default-format>;

Items in italics can also be the generic name for a list of arguments from which you can choose (for example, attribute-list). If more than one of an item in italics can be used, the items are expressed as item-1, ..., item-n.

**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 the following example of the MAPS system option, the equal sign sets the value of MAPS:
    
    MAPS = location-of-maps

- **<>**
  - angle brackets identify optional arguments. Any argument that is not enclosed in angle brackets is required.
  - In the following example of the CAT function, at least one item is required:
    
    CAT (item-1 <, ..., item-n>)

- **|**
  - 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 the following example of the CMPMODEL= system option, you can choose only one of the arguments:
    
    CMPMODEL = BOTH | CATALOG | XML

- **...**
  - an ellipsis indicates that the argument or group of arguments following the ellipsis can be repeated. If the ellipsis and the following argument are enclosed in angle brackets, then the argument is optional.
  - In the following example of the CAT function, the ellipsis indicates that you can have multiple optional items:
    
    CAT (item-1 <, ..., item-n>)

- **'value' or “value”**
  - indicates that an argument enclosed in single or double quotation marks must have a value that is also enclosed in single or double quotation marks.
  - In the following 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 the following 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 usually have a choice of using a SAS statement (LIBNAME or FILENAME) or the operating environment's control language to make the association. 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`. Note that `SAS-library` is enclosed in quotation marks:

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```
What’s New in SAS 9.4 Data Set Options

Overview

SAS 9.4 Data Set Options includes one new data set option and two enhanced data set options.

- The new ENCRYPTKEY= on page 23 data set option specifies a key value. This enables you to use AES (Advanced Encryption Standard) encryption.

- The enhanced ENCRYPT= on page 18 data set option now supports AES encryption. This enables you to use the AES algorithm for stronger encryption.

- The default for the EXTENDOBS_COUNTER= on page 26 data set option is now YES, which creates a SAS data file with an enhanced file format. The enhanced file format extends the observation count beyond the 32-bit long limitation.
Chapter 1
About Data Set Options

Definition of Data Set Options

Data set options specify actions that apply only to the SAS data set with which they appear. Data set options enable you to perform these operations:

- rename variables
- select only the first or last \( n \) observations for processing
- drop variables from processing or from the output data set
- specify a password for a data set

Syntax

Specify a data set option in parentheses after a SAS data set name. To specify several data set options, separate them with spaces.

\((\text{option-}1=\text{value-}1<...\text{option-n}=\text{value-n}>\))

The following examples show data set options in SAS statements:

- `data scores(keep=team game1 game2 game3);`
- `data mydata(index=(b k) label='label for my data set' drop=p read=secret);`
- `data new(drop=i n index=(j combo=(x1 a1 a20 b1 b50 )));`
- `data idxdup2(compress=yes index=(ok1 ok2 ssn/unique ok3));`
- `proc print data=new(drop=year);`
Using Data Set Options

Using Data Set Options with Input or Output SAS Data Sets

Most SAS data set options apply to either input or output SAS data sets in DATA steps or procedure (PROC) steps. If a data set option is associated with an input data set, the action applies to the data set that is being read. If the data set option appears in the DATA statement or after an output data set specification in a PROC step, SAS applies the action to the output data set. In the DATA step, data set options for output data sets must appear in the DATA statement, not in any OUTPUT statements that might be present.

Some data set options, such as COMPRESS=, are meaningful only when you create a SAS data set because they set attributes that exist for the duration of the data set. To change or cancel most data set options, you must re-create the data set. You can change other options (such as PW= and LABEL=) with PROC DATASETS. For more information, see “DATASETS” in Base SAS Procedures Guide.

When data set options appear in input and output data sets in the same DATA or PROC step, SAS first applies data set options to input data sets. Then SAS evaluates programming statements or applies data set options to output data sets. Likewise, data set options that are specified for the data set that is being created are applied after programming statements are processed. For example, when you are using the RENAME= data set option, the new names are not associated with the variables until the DATA step ends.

In some instances, data set options conflict when they are used in the same statement. For example, you cannot specify the DROP= and KEEP= data set options for the same variable in the same statement. Timing can also be an issue in some cases. For example, if you are using KEEP= and RENAME= on a data set that is specified in the SET statement, KEEP= needs to use the original variable names. SAS processes KEEP= before the data set is read. The new names that are specified in RENAME= apply to the programming statements that follow the SET statement.

How Data Set Options Interact with System Options

Many system options and data set options share the same name and have the same function. System options remain in effect for all DATA steps and PROC steps in a SAS job or session.

The data set option overrides the system option for the data set in the step in which it appears. In this example, the OBS= system option in the OPTIONS statement specifies that only the first 100 observations are processed from any data set within the SAS job. However, the OBS= data set option in the SET statement overrides the system option for data set TWO. OBS= specifies that only the first five observations are read from data set TWO. The PROC PRINT step prints the data set FINAL. This data set contains the first five observations from data set TWO, followed by the first 100 observations from data set THREE:

```plaintext
options obs=100;

data final;
```
Data Set Options Documented in Other SAS Publications

In addition to data set options documented in *SAS Data Set Options: Reference*, data set options are also documented in the following publications:

- *SAS Companion for Windows*
- *SAS Companion for UNIX Environments*
- *SAS Companion for z/OS*
- *SAS Scalable Performance Data Engine: Reference*
- *SAS/ACCESS for Relational Databases: Reference*
- *Base SAS Procedures Guide*
Chapter 2
Dictionary of Data Set Options

Data Set Options by Category

Dictionary

ALTERN= Data Set Option
BUFFNO= Data Set Option
BUFFSIZE= Data Set Option
CNTLLEV= Data Set Option
COMPRESS= Data Set Option
DLDMGACTION= Data Set Option
DROP= Data Set Option
ENCRYPT= Data Set Option
ENCRYPTKEY= Data Set Option
EXTENDOBSCOUNTER= Data Set Option
FILECLOSE= Data Set Option
FIRSTOBS= Data Set Option
GENMAX= Data Set Option
GENNUM= Data Set Option
IDXNAME= Data Set Option
IDXWHERE= Data Set Option
INDEX= Data Set Option
IN= Data Set Option
KEEP= Data Set Option
LABEL= Data Set Option
OBSBUF= Data Set Option
OBS= Data Set Option
OUTREP= Data Set Option
POINTOBS= Data Set Option
PW= Data Set Option
PWREQ= Data Set Option
READ= Data Set Option
RENAME= Data Set Option
REPEMPTY= Data Set Option
REPLACE= Data Set Option
REUSE= Data Set Option
ROLE= Data Set Option
SORTEDBY= Data Set Option
SPILL= Data Set Option
TOBSNO= Data Set Option
TYPE= Data Set Option
WHERE= Data Set Option
WHEREUP= Data Set Option
WRITE= Data Set Option
## Data Set Options by Category

The categories for SAS data set options correspond to the SAS data set option groups:

- **Data Set Control**: options that are associated with data sets
- **Observation Control**: options that are associated with observations
- **User Control of SAS Index Usage**: options that are associated with indexes
- **Variable Control**: options that are associated with variables
- **Miscellaneous**: option that is associated with tape position

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Control</td>
<td>ALTER= Data Set Option (p. 8)</td>
<td>Assigns an ALTER= password to a SAS file that prevents users from replacing or deleting the file, and enables access to a read-and-write-protected file.</td>
</tr>
<tr>
<td></td>
<td>BUFNO= Data Set Option (p. 9)</td>
<td>Specifies the number of buffers to be allocated for processing a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>BUFSIZE= Data Set Option (p. 11)</td>
<td>Specifies the size of a permanent buffer page for an output SAS data set.</td>
</tr>
<tr>
<td></td>
<td>CNTLLEV= Data Set Option (p. 12)</td>
<td>Specifies the level of shared access to a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>COMPRESS= Data Set Option (p. 13)</td>
<td>Specifies how observations are compressed in a new output SAS data set.</td>
</tr>
<tr>
<td></td>
<td>DLDIMGACTION= Data Set Option (p. 16)</td>
<td>Specifies the action to take when a SAS data set in a SAS library is detected as damaged.</td>
</tr>
<tr>
<td></td>
<td>ENCRYPT= Data Set Option (p. 18)</td>
<td>Specifies whether to encrypt an output SAS data set.</td>
</tr>
<tr>
<td></td>
<td>ENCRYPTKEY= Data Set Option (p. 23)</td>
<td>Specifies a key value for AES (Advanced Encryption Standard) encryption.</td>
</tr>
<tr>
<td></td>
<td>EXTENDOBS= Data Set Option (p. 26)</td>
<td>Specifies whether to extend the maximum observation count in a new output SAS data file.</td>
</tr>
<tr>
<td></td>
<td>GENMAX= Data Set Option (p. 29)</td>
<td>Requests generations for a new data set, modifies the number of generations for an existing data set, and specifies the maximum number of versions.</td>
</tr>
<tr>
<td></td>
<td>GENNUM= Data Set Option (p. 31)</td>
<td>Specifies a particular generation of a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>INDEX= Data Set Option (p. 35)</td>
<td>Defines an index for a new output SAS data set.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>LABEL= Data Set Option (p. 38)</td>
<td>Specifies a label for a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>OBSBUF= Data Set Option (p. 40)</td>
<td>Determines the size of the view buffer for processing a DATA step view.</td>
</tr>
<tr>
<td></td>
<td>OUTREP= Data Set Option (p. 50)</td>
<td>Specifies the data representation for the output SAS data set.</td>
</tr>
<tr>
<td></td>
<td>PW= Data Set Option (p. 54)</td>
<td>Assigns a READ, WRITE, and ALTER password to a SAS file, and enables access to a password-protected SAS file.</td>
</tr>
<tr>
<td></td>
<td>PWREQ= Data Set Option (p. 55)</td>
<td>Specifies whether to display a password dialog box.</td>
</tr>
<tr>
<td></td>
<td>READ= Data Set Option (p. 55)</td>
<td>Assigns a READ= password to a SAS file that prevents users from reading the file, unless they enter the password.</td>
</tr>
<tr>
<td></td>
<td>REPEMPTY= Data Set Option (p. 59)</td>
<td>Specifies whether a new, empty data set can overwrite an existing SAS data set that has the same name.</td>
</tr>
<tr>
<td></td>
<td>REPLACE= Data Set Option (p. 60)</td>
<td>Specifies whether a new SAS data set that contains data can overwrite an existing data set that has the same name.</td>
</tr>
<tr>
<td></td>
<td>REUSE= Data Set Option (p. 61)</td>
<td>Specifies whether new observations can be written to available space in compressed SAS data sets.</td>
</tr>
<tr>
<td></td>
<td>ROLE= Data Set Option (p. 62)</td>
<td>Identifies the fact table for a star schema join.</td>
</tr>
<tr>
<td></td>
<td>SORTEDBY= Data Set Option (p. 63)</td>
<td>Specifies how a data set is currently sorted.</td>
</tr>
<tr>
<td></td>
<td>SPILL= Data Set Option (p. 65)</td>
<td>Specifies whether to create a spill file for non-sequential processing of a DATA step view.</td>
</tr>
<tr>
<td></td>
<td>TOBSNO= Data Set Option (p. 73)</td>
<td>Specifies the number of observations to send in a client/server transfer.</td>
</tr>
<tr>
<td></td>
<td>TYPE= Data Set Option (p. 73)</td>
<td>Specifies the data set type for a specially structured SAS data set.</td>
</tr>
<tr>
<td></td>
<td>WRITE= Data Set Option (p. 78)</td>
<td>Assigns a WRITE= password to a SAS file that prevents users from writing to a file, unless the users enter the password.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>FILECLOSE= Data Set Option (p. 27)</td>
<td>Specifies how a tape is positioned when a SAS data set is closed.</td>
</tr>
<tr>
<td>Observation</td>
<td>FIRSTOBS= Data Set Option (p. 28)</td>
<td>Specifies the first observation that SAS processes in a SAS data set.</td>
</tr>
<tr>
<td>Control</td>
<td>IN= Data Set Option (p. 36)</td>
<td>Creates a Boolean variable that indicates whether the data set contributed data to the current observation.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>OBS= Data Set Option (p. 41)</td>
<td>Specifies the last observation that SAS processes in a data set.</td>
</tr>
<tr>
<td></td>
<td>POINTOBS= Data Set Option (p. 52)</td>
<td>Specifies whether SAS creates compressed data sets whose observations can be randomly accessed or sequentially accessed.</td>
</tr>
<tr>
<td></td>
<td>WHERE= Data Set Option (p. 74)</td>
<td>Specifies specific conditions to use to select observations from a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>WHERESP= Data Set Option (p. 76)</td>
<td>Specifies whether to evaluate new observations and modified observations against a WHERE expression.</td>
</tr>
<tr>
<td>User Control of SAS Index Usage</td>
<td>IDXNAME= Data Set Option (p. 32)</td>
<td>Directs SAS to use a specific index to match the conditions of a WHERE expression.</td>
</tr>
<tr>
<td></td>
<td>IDXWHERE= Data Set Option (p. 33)</td>
<td>Specifies whether SAS uses an index search or a sequential search to match the conditions of a WHERE expression.</td>
</tr>
<tr>
<td>Variable Control</td>
<td>DROP= Data Set Option (p. 17)</td>
<td>For an input data set, excludes the specified variables from processing; for an output data set, excludes the specified variables from being written to the data set.</td>
</tr>
<tr>
<td></td>
<td>KEEP= Data Set Option (p. 37)</td>
<td>For an input data set, specifies the variables to process; for an output data set, specifies the variables to write to the data set.</td>
</tr>
<tr>
<td></td>
<td>RENAME= Data Set Option (p. 56)</td>
<td>Changes the name of a variable.</td>
</tr>
</tbody>
</table>

## Dictionary

### ALTER= Data Set Option

Assigns an ALTER= password to a SAS file that prevents users from replacing or deleting the file, and enables access to a read- and write-protected file.

**Valid in:** DATA step and PROC steps  
**Category:** Data Set Control  
**Note:** Check your log after this operation to ensure password security. For more information, see “Blotting Passwords and Encryption Key Values” in SAS Language Reference: Concepts.  
**See:** ALTER= Data Set Option under OpenVMS, UNIX, or z/OS in the documentation for your operating environment.

**Syntax**

```
ALTER=alter-password
```
Syntax Description

*alter-password*

must be a valid SAS name. For more information, see “Words in the SAS Language” in *SAS Language Reference: Concepts*.

Details

The ALTER= option applies to all types of SAS files except catalogs. You can use this option to assign a password to a SAS file or to access a read-protected, write-protected, or alter-protected SAS file.

When replacing a SAS data set that is protected with an ALTER password, the new data set inherits the ALTER password. To change the ALTER password for the new data set, use the MODIFY statement in the DATASETS procedure.

*Note:* A SAS password does not control access to a SAS file beyond the SAS system.

Use the operating system-supplied utilities and file-system security controls in order to control access to SAS files outside of SAS.

See Also

- “File Protection” in *SAS Language Reference: Concepts*
- “Manipulating Passwords” in *Base SAS Procedures Guide*

**Data Set Options:**

- “ENCRYPT= Data Set Option” on page 18
- “PW= Data Set Option” on page 54
- “READ= Data Set Option” on page 55
- “WRITE= Data Set Option” on page 78

---

**BUFNO= Data Set Option**

Specifies the number of buffers to be allocated for processing a SAS data set.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step and PROC steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Data Set Control</td>
</tr>
<tr>
<td>See:</td>
<td>BUFNO= Data Set Option in the documentation for your operating environment.</td>
</tr>
</tbody>
</table>

**Syntax**

```
BUFNO= n | nK | hexX | MIN | MAX
```

**Syntax Description**

*n | nK*

specifies the number of buffers in multiples of 1 (bytes); 1,024 (kilobytes). For example, a value of 8 specifies 8 buffers, and a value of 1k specifies 1,024 buffers.
hexX specifies the number of buffers as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the number of buffers to 45.

MIN sets the minimum number of buffers to 0, which causes SAS to use the minimum optimal value for the operating environment. This is the default.

MAX sets the number of buffers to the maximum possible number in your operating environment, up to the largest four-byte, signed integer, which is $2^{31} - 1$, or approximately 2 billion.

**Details**

The buffer number is not a permanent attribute of the data set; it is valid only for the current SAS session or job.

**BUFNO=** applies to SAS data sets that are opened for input, output, or update.

A larger number of buffers can speed execution time by limiting the number of input and output (I/O) operations that are required for a particular SAS data set. However, the improvement in execution time comes at the expense of increased memory consumption.

To reduce I/O operations on a small data set as well as speed execution time, allocate one buffer for each page of data to be processed. This technique is most effective if you read the same observations several times during processing.

To request that SAS allocate the number of buffers based on the number of data set pages and index file pages, use the SASFILE global statement.

**Operating Environment Information**

The default value for BUFNO= is determined by your operating environment and is set to optimize sequential access. To improve performance for direct (random) access, change the value for BUFNO=. For the default setting and possible settings for direct access, see the BUFNO= data set option in the SAS documentation for your operating environment.

**Comparisons**

- If the BUFNO= data set option is not specified, then the value of the BUFNO= system option is used. If both are specified in the same SAS session, the value that is specified for the BUFNO= data set option overrides the value that is specified for the BUFNO= system option.

**See Also**

**Data Set Options:**
- “BUFSIZE= Data Set Option” on page 11

**System Options:**
- “BUFNO= System Option” in *SAS System Options: Reference*

**Statements:**
- “SASFILE Statement” in *SAS Statements: Reference*
BUFSIZE= Data Set Option

Specifies the size of a permanent buffer page for an output SAS data set.

Valid in: DATA step and PROC steps
Category: Data Set Control
Restriction: Use with output data sets only.
See: BUFSIZE= Data Set Option under UNIX, z/OS, or OpenVMS in the documentation for your operating environment.

Syntax

BUFSIZE=n | nK | nM | nG | hexX | MAX

Syntax Description

n | nK | nM | nG
specifies the page size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). For example, a value of 8 specifies a page size of 8 bytes, and a value of 4k specifies a page size of 4,096 bytes.

Note: If the system option and the data set option are not set, the default is 0. As a result, SAS uses the minimum optimal page size for the operating environment. The BUFSIZE= system option is used in either of the following scenarios:
• if the BUFSIZE= data set option is not set
• if the BUFSIZE= data set option is set to zero
Use BUFSIZE=0 to reset the buffer page size to the default value in your operating environment.

hexX
specifies the page size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the page size to 45 bytes.

MAX
sets the page size to the maximum possible number in your operating environment, up to the largest four-byte, signed integer, which is $2^{31}-1$, or approximately 2 billion bytes.

Details

The page size is the amount of data that can be transferred for a single I/O operation to one buffer. The page size is a permanent attribute of the data set and is used when the data set is processed.

A larger page size can speed execution time by reducing the number of times SAS has to read from or write to the storage medium. However, the improvement in execution time comes at the expense of increased memory consumption.

To change the page size, use a DATA step to copy the data set and either specify a new page or use the SAS default. To reset the page size to the default value in your operating environment, use BUFSIZE=0.
Note: You can use the COPY procedure to copy a data set to another library that is allocated with a different engine. The specified page size of the data set is not retained.

Operating Environment Information
The default value for BUFSIZE= is determined by your operating environment and is set to optimize sequential access. To improve performance for direct (random) access, change the value for BUFSIZE=. For the default setting and possible settings for direct access, see the BUFSIZE= data set option in the SAS documentation for your operating environment.

See Also

Data Set Options:
• “BUFNO= Data Set Option” on page 9

System Options:
• “BUFSIZE= System Option” in SAS System Options: Reference

CNTLLEV= Data Set Option
Specifies the level of shared access to a SAS data set.

Valid in: DATA step and PROC steps
Category: Data Set Control
Restriction: Specify for input data sets only.

Syntax
CNTLLEV=LIB | MEM | REC

Syntax Description
LIB
specifies that concurrent access is controlled at the library level. Library-level control restricts concurrent access to only one update process to the library.

MEM
specifies that concurrent access is controlled at the SAS data set (member) level. Member-level control restricts concurrent access to only one update or output process to the SAS data set. If the data set is open for an update or output process, then no other operation can access the data set. If the data set is open for an input process, then other concurrent input processes are allowed, but no update or output process is allowed.

REC
specifies that concurrent access is controlled at the observation (record) level. Record-level control allows more than one Update access to the same SAS data set, but it denies concurrent update of the same observation.
Details

The CNTLLEV= option specifies the level at which shared Update access to a SAS data set is denied. A SAS data set can be opened concurrently by more than one SAS session or by more than one statement, window, or procedure within a single session. By default, SAS procedures allow the greatest degree of concurrent access possible, and they guarantee the integrity of the data and the data analysis. Therefore, you typically use the CNTLLEV= data set option in these situations:

• when your application controls the access to the data, such as in SAS Component Language (SCL), SAS/IML software, or DATA step programming
• when you access data through an interface engine that does not provide member-level control of the data.

If you use CNTLLEV=REC and the SAS procedure needs member-level control for integrity of the data analysis, SAS prints a warning to the SAS log. The warning states that inaccurate or unpredictable results can occur if the data is updated by another process during the analysis.

Example: Changing the Shared Access Level

In this example, the first SET statement includes the CNTLLEV= data set option in order to override the default level of shared access from member-level control to record-level control. The second SET statement opens the SAS data set with the default member-level control.

```sas
set datalib.fuel (cntllev=rec) point=obsnum;
.
.
set datalib.fuel;
by area;
```

COMPRESS= Data Set Option

Specifies how observations are compressed in a new output SAS data set.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step and PROC steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Data Set Control</td>
</tr>
</tbody>
</table>
| Restriction: Use with output data sets only.

Syntax

COMPRESS= NO | YES | CHAR | BINARY

Syntax Description

NO
specifies that the observations in a newly created SAS data set are uncompressed (maintaining fixed-length records).

YES | CHAR
specifies that the observations in a newly created SAS data set are compressed (producing variable-length records) by using RLE (Run Length Encoding). RLE
compresses observations by reducing repeated runs of the same character (including blanks) to two-byte or three-byte representations.

Alias ON

**BINARY**

specifies that the observations in a newly created SAS data set are compressed (producing variable-length records) by using RDC (Ross Data Compression). RDC combines run-length encoding and sliding-window compression to compress the file by representing repeated byte patterns more efficiently.

*Note:* This method is highly effective for compressing medium to large (several hundred bytes or larger) blocks of binary data (character and numeric variables). Because the compression function operates on a single record at a time, the record length needs to be several hundred bytes or larger for effective compression.

**Details**

Compressing a file reduces the number of bytes that are required to represent each observation. Advantages of compressing a file include reduced storage requirements for the file and fewer I/O operations to read or write to the data during processing. However, more CPU resources are required to read a compressed file (because of the overhead of uncompressing each observation). There are situations where the resulting file size might increase rather than decrease.

Use the COMPRESS= data set option to compress an individual file. Specify the option for output data sets only. That is, specify data sets named in the DATA statement of a DATA step or in the OUT= option of a SAS procedure. Use the COMPRESS= data set option only when you are creating a SAS data file (member type DATA). You cannot compress SAS views, because they contain no data. The COPY procedure does not support data set options. Therefore, you cannot use the COMPRESS= data set option in PROC COPY or a COPY statement from PROC DATASETS.

**Tip** To compress an OUTPUT data set that is generated by PROC COPY, you can use the COMPRESS=YES system option before the PROC COPY statement with the NOCLONE option.

```plaintext
options compress=yes;
proc copy in=work out=new noclone;
select x;
run;
```

After a file is compressed, the setting is a permanent attribute of the file. To change the setting, you must re-create the file. That is, to uncompress a file, specify COMPRESS=NO for a DATA step that copies the compressed file.

In general, COMPRESS=CHAR provides good compression when single bytes repeat; COMPRESS=BINARY provides good compression when strings of bytes repeat. It is more costly to look for strings of bytes that repeat, than to look for single bytes that repeat. For examples, see “Example 1: COMPRESS=CHAR” on page 15 and “Example 2: COMPRESS=BINARY” on page 15.

**Comparisons**

The COMPRESS= data set option overrides the COMPRESS= option in the LIBNAME statement and the COMPRESS= system option.
The data set option POINTOBS=YES, which is the default, determines that a compressed data set can be processed with random access (by observation number) rather than sequential access. With random access, you can specify an observation number in the FSEDIT procedure and the POINT= option in the SET and MODIFY statements.

When you create a compressed file, you can also specify REUSE=YES (as a data set option or system option) to track and reuse space. With REUSE=YES, new observations are inserted in available space when other observations are updated or deleted. When the default REUSE=NO is in effect, new observations are appended to the existing file.

POINTOBS=YES and REUSE=YES are mutually exclusive. That is, they cannot be used together. REUSE=YES takes precedence over POINTOBS=YES. If you set REUSE=YES, SAS automatically sets POINTOBS=NO.

The TAPE engine supports the COMPRESS= data set option, but the engine does not support the COMPRESS= system option.

The XPORT engine does not support compression.

Examples

Example 1: COMPRESS=CHAR

```sas
data mylib.CharRepeats(compress=char);
  length ca $ 200;
  do i=1 to 100000;
    ca='aaaaaaaaaaaaaaaaaaaaaa';
    cb='bbbbbbbbbbbbbbbbbbbbbb';
    cc='cccccccccccccccccccccc';
    output;
  end;
run;
```

The following message is written to the SAS log:

```
NOTE: Compressing data set MYLIB.CHARREPEATS decreased size by 88.55 percent.
Compressed is 45 pages; un-compressed would require 393 pages.
```

Example 2: COMPRESS=BINARY

```sas
data mylib.StringRepeats(compress=binary);
  length cabcd $ 200;
  do i=1 to 1000000;
    cabcd='abcdabcdabcdabcdabcd';
    cefgh='efghgfhefhefhegheghefgh';
    cijkl='ijklijklijklijklijkl';
    output;
  end;
run;
```

The following message is written to the SAS log:

```
NOTE: Compressing data set MYLIB.STRINGREPEATS decreased size by 70.27 percent.
Compressed is 1239 pages; un-compressed would require 4167 pages.
```

See Also

- “Compressing Data Files” in SAS Language Reference: Concepts
Data Set Options:
• “POINTOBS= Data Set Option” on page 52
• “REUSE= Data Set Option” on page 61

Statements:
• “LIBNAME Statement” in SAS Statements: Reference

System Options:
• “COMPRESS= System Option” in SAS System Options: Reference
• “REUSE= System Option” in SAS System Options: Reference

DLDMGACTION= Data Set Option
Specifies the action to take when a SAS data set in a SAS library is detected as damaged.

Valid in: DATA step and PROC steps
Category: Data Set Control
Defaults: For Windows and UNIX, the shipped default is REPAIR for interactive mode and FAIL for batch mode.
For z/OS, the shipped default is PROMPT for interactive mode and REPAIR for batch mode.

Syntax
DLDMGACTION=FAIL | ABORT | REPAIR | NOINDEX | PROMPT

Syntax Description
FAIL
stops the step and issues an error message to the log immediately. This is the default for batch mode.
ABORT
stops the step, issues an error message to the log, and terminates the SAS session.
REPAIR
attempts to automatically repair a damaged data set on the next attempt to open the damaged file. The data set might truncate at the point of damage. The REPAIR option re-creates the index or indexes. If the damage is too severe, the auto repair attempt might not be successful.
NOINDEX
automatically repairs the data file without the indexes and integrity constraints. The repair also deletes the index file and updates the data file to reflect the disabled indexes and integrity constraints. The repair limits the data file to be opened only in INPUT mode. A warning is written to the SAS log instructing you to execute the PROC DATASETS REBUILD statement to correct or delete the disabled indexes and integrity constraints.

See “DLDMGACTION= System Option” in SAS System Options: Reference
PROMPT
displays a dialog box that asks you to select the FAIL, ABORT, REPAIR, or NOINDEX action.

DROP= Data Set Option
For an input data set, excludes the specified variables from processing; for an output data set, excludes the specified variables from being written to the data set.

Valid in: DATA step and PROC steps
Category: Variable Control

Syntax
DROP=variable(s)

Syntax Description
variable(s)
list one or more variable names. You can list the variables in any form that SAS allows.

Details
If the option is associated with an input data set, the variables are not available for processing. If the DROP= data set option is associated with an output data set, SAS does not write the variables to the output data set, but they are available for processing.

Comparisons
The DROP= data set option differs from the DROP statement in these ways:
• In DATA steps, the DROP= data set option can apply to input and output data sets. The DROP statement applies only to output data sets.
• In DATA steps, when you create multiple output data sets, use the DROP= data set option to write different variables to different data sets. The DROP statement applies to all output data sets.
• In PROC steps, you can use only the DROP= data set option, not the DROP statement.

Examples

Example 1: Excluding Variables from Input
In this example, the variables SALARY and GENDER are not included in processing, and they are not written to either output data set:

```sas
data plan1 plan2;
  set payroll(drop=salary gender);
  if hired<'01jan98'd then output plan1;
```
You cannot use SALARY or GENDER in any logic in the DATA step because DROP= prevents the SET statement from reading SALARY and GENDER from PAYROLL.

**Example 2: Processing Variables without Writing Them to a Data Set**

In this example, SALARY and GENDER are not written to PLAN2, but they are written to PLAN1:

```sas
   data plan1 plan2(drop=salary gender);
   set payroll;
   if hired<='01jan98'd then output plan1;
   else output plan2;
   run;
```

**See Also**

**Data Set Options:**
- “KEEP= Data Set Option” on page 37

**Statements:**
- “DROP Statement” in *SAS Statements: Reference*

---

**ENCRYPT= Data Set Option**

Specifies whether to encrypt an output SAS data set.

**Valid in:** DATA step and PROC steps  
**Category:** Data Set Control  
**Default:** ENCRYPT=NO  
**Restriction:** Use with output data sets only.

**Syntax**

```sas
ENCRYPT=AES | NO | YES
```

**Syntax Description**

**AES**

encrypts the file by using the AES (Advanced Encryption Standard) algorithm. AES provides enhanced encryption by using SAS/SECURE software, which is included with Base SAS software. You must specify the ENCRYPTKEY= data set option when you are using ENCRYPT=AES. For more information, see “ENCRYPTKEY= Data Set Option” on page 23.

**Restriction**

The tape engine does not support ENCRYPT=AES. Use ENCRYPT=YES for tape engine encryption.

**CAUTION**

Record all ENCRYPTKEY= values when you are using ENCRYPT=AES. If you forget to record the ENCRYPTKEY= value,
you lose your data. SAS cannot assist you in recovering the ENCRYPTKEY= value. The following note is written to the log:

Note: If you lose or forget the ENCRYPTKEY= value, there will be no way to open the file or recover the data.

NO
does not encrypt the file.

YES
encrypts the file by using the SAS Proprietary algorithm. This encryption uses passwords that are stored in the data set. At a minimum, you must specify the READ= data set option or the PW= data set option at the same time that you specify ENCRYPT=Yes. Because the encryption method uses passwords, you cannot change any password on an encrypted data set without re-creating the data set.

CAUTION:
Record all passwords when you are using ENCRYPT=YES. If you forget the passwords, you cannot reset them without assistance from SAS. This is a time-consuming and resource-intensive process.

Details
To use ENCRYPT=YES data files, you must have SAS 6.11 or later. When you use ENCRYPT=YES, these rules apply:

• To copy an encrypted data file, the output engine must support the encryption.
• If the data file is encrypted, all associated indexes are also encrypted.
• You cannot use PROC CPORT on SASProprietary encrypted data files.

Note: SAS views do not contain data. Therefore, encryption is not necessary.

To use encrypted AES data files, you must use SAS 9.4 or later and SAS/SECURE software. In addition, when you use ENCRYPT=AES, these rules apply:

• You must use the ENCRYPTKEY= data set option when creating a data set with AES encryption.
• To copy an encrypted AES data file, the output engine must support AES encryption.
• In Base SAS, data files with referential integrity constraints can use AES encryption. All primary key and foreign key data files must use the same encryption key that opens all referencing foreign key and primary key data files.

You cannot change the ENCRYPTKEY= value on an AES-encrypted data file without re-creating the data file.

Note: Encryption requires approximately the same amount of CPU resources as compression.

Examples

Example 1: Using the ENCRYPT=Yes Option
The following example encrypts the data set by using the SAS Proprietary algorithm:

```
libname mylib "c:\mylib";

data mylib.salary(encrypt=yes read=green);
```
input name $ yrsal bonuspct;
datalines;
Muriel  34567  3.2
Bjorn   74644  2.5
Freda   38755  4.1
Benny   29855  3.5
Agnetha 70998  4.1
;

To use this data set, specify the READ= password:

proc contents data=mylib.salary(read=green);
run;


Example 2: Using the ENCRYPT=AES Option

```sas
libname mylib "c:\mylib";

data mylib.salary(encrypt=aes encryptkey=green);
  input name $ yrsal bonuspct;
  datalines;
Muriel    34567  3.2
Bjorn     74644  2.5
Freda     38755  4.1
Benny     29855  3.5
```
The following example encrypts the data set by using the AES algorithm:

```sas
proc contents data=mylib.salary(encryptkey=green);
run;
```

Output 2.2  ENCRYPT=AES

---

## The SAS System

### The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>MYLIB.SALARY</th>
<th>Observations</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>3</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>04/30/2014 13:42:52</td>
<td>Observation Length</td>
<td>24</td>
</tr>
<tr>
<td>Last Modified</td>
<td>04/30/2014 13:42:52</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
<td>Compressed</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td>AES</td>
<td>Sorted</td>
<td>NO</td>
</tr>
<tr>
<td>Encrypted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>WINDOWS_64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>w\latin1 Western (Windows)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Engine/Host Dependent Information

<table>
<thead>
<tr>
<th>Data Set Page Size</th>
<th>65536</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Data Set Pages</td>
<td>1</td>
</tr>
<tr>
<td>First Data Page</td>
<td>1</td>
</tr>
<tr>
<td>Max Obs per Page</td>
<td>2715</td>
</tr>
<tr>
<td>Obs in First Data Page</td>
<td>5</td>
</tr>
<tr>
<td>Number of Data Set Repairs</td>
<td>0</td>
</tr>
<tr>
<td>ExtendObsCounter</td>
<td>YES</td>
</tr>
<tr>
<td>Filename</td>
<td>c:\mylib\salary.sas7bdat</td>
</tr>
<tr>
<td>Release Created</td>
<td>9.0401MO</td>
</tr>
<tr>
<td>Host Created</td>
<td>X64_7PRO</td>
</tr>
</tbody>
</table>
See Also

- “SAS Data File Encryption” in *SAS Language Reference: Concepts*

Data Set Options:

- “ALTER= Data Set Option” on page 8
- “PW= Data Set Option” on page 54
- “READ= Data Set Option” on page 55
- “WRITE= Data Set Option” on page 78

---

**ENCRYPTKEY= Data Set Option**

Specifies a key value for AES (Advanced Encryption Standard) encryption.

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Range:** 1 to 64 bytes

**Restrictions:** Use with SAS 9.4 or later only.
Use only with AES-encrypted data files.

**Note:** Check your log after this operation to ensure encryption key security. For more information, see “Blotting Passwords and Encryption Key Values” in *SAS Language Reference: Concepts.*

---

**Syntax**

`ENCRYPTKEY=key-value`

**Syntax Description**

`key-value`

assigns an encrypt key value. You must specify the ENCRYPTKEY= data set option when you are using ENCRYPT=AES. The key value can be up to 64 bytes long. To create an ENCRYPTKEY= key value with or without quotation marks, follow these rules:

**No quotation marks:**

- use alphanumeric characters and underscores only
- can be up to 64 bytes long
- use uppercase and lowercase letters
- must start with a letter
- cannot include blank spaces
- is not case sensitive

```sas
%let mykey=abcdefghi12;
encryptkey=&mykey
encryptkey=key_value
encryptkey=key_value1
```
Single quotation marks:
- use alphanumeric, special, and DBCS characters
- can be up to 64 bytes long
- use uppercase and lowercase letters
- can include blank spaces, but cannot contain all blanks
- is case sensitive

```
encryptkey='key_value'
encryptkey='1234*#mykey'
```

Double quotation marks:
- use alphanumeric, special, and DBCS characters
- can be up to 64 bytes long
- use uppercase and lowercase letters
- can include blank spaces, but cannot contain all blanks
- is case sensitive

```
encryptkey="key_value"
encryptkey="1234*#mykey"
%let mykey=Abcdefghi12;
encryptkey="&mykey"
```

When the ENCRYPTKEY= key value uses DBCS characters, the 64-byte limit applies to the character string after it has been transcoded to UTF-8 encoding. You can use the following DATA step to calculate the length in bytes of a key value in DBCS:

```
data _null_
   key=length(unicodec('key-value','UTF8'));
   put 'key length=' key;
run;
```

Interaction
You cannot change the key value on an AES-encrypted data set without re-creating the data set.

Details

**CAUTION:**
Record all ENCRYPTKEY= values when you are using ENCRYPT=AES. If you forget to record the ENCRYPTKEY= value, you lose your data. SAS cannot assist you in recovering the ENCRYPTKEY= value.

You must use the ENCRYPTKEY= option when you are creating or accessing a SAS data set with AES encryption.

The ENCRYPTKEY= data set option does not protect the file from deletion or replacement. Encrypted data sets can be deleted using any of the following scenarios without specifying an ENCRYPTKEY= key value:

- the KILL option in PROC DATASETS
- the DROP statement in PROC SQL
- the DELETE procedure
The ENCRYPTKEY= option only prevents access to the contents of the file. To protect the file from deletion or replacement, the file must also contain an ALTER= password.

You must specify the ENCRYPTKEY= key value when you copy AES-encrypted data files. The value follows the data set name in the SELECT statement. The following example uses the SELECT statement:

```sas
   copy in=OldLib out=NewLib;
   select salary(encryptkey=key-value);
   run;
```

When working with data files that are protected with the ENCRYPTKEY= key value in the DATASETS procedure, you can specify the value in the AGE, APPEND, AUDIT, CONTENTS, MODIFY, REBUILD, and REPAIR statements. You must also specify the value when the CHANGE statement refers to a specific generation data set by using a relative reference to the value:

```sas
   change OldName(gennum=-1 encryptkey=key-value)=NewName;
   run;
```

The option can be specified either in parentheses after the name of the SAS data file or after a forward slash.

**CAUTION:**

When you are using referential integrity constraints, all primary key and foreign key data files that reference each other must use the same encryption key. For more information, see “Encryption and Integrity Constraints” in *SAS Language Reference: Concepts*.

You can use a macro variable as the ENCRYPTKEY= key value. The following code defines a macro variable:

```sas
   %let secret=myvalue;
```

The following code uses the macro variable as the ENCRYPTKEY= key value:

```sas
   data my.dsname(encrypt=aes encryptkey="&secret");
```

When you specify a macro variable for the ENCRYPTKEY= key value, you must enclose the macro variable in double quotation marks. If you do not use double quotation marks, unpredictable results can occur.

**Example: Using the ENCRYPTKEY= Option**

This example uses the ENCRYPT=AES option:

```sas
   data salary(encrypt=aes encryptkey=green);
   input name $ yrsal bonuspct;
   datalines;
   Muriel  34567  3.2
   Bjorn   74644  2.5
   Freda   38755  4.1
   Benny   29855  3.5
   Agnetha 70998  4.1
   proc contents data=salary(encryptkey=green);
   run;
```
EXTENDOBSCOUNTER= Data Set Option

Specifies whether to extend the maximum observation count in a new output SAS data file.

- **Valid in:** DATA step and PROC steps
- **Category:** Data Set Control
- **Alias:** EOC=
- **Default:** YES
- **Restrictions:**
  - Use with output data files only.
  - Use with the BASE engine only.

### Syntax

**EXTENDOBSCOUNTER=**YES | NO

### Syntax Description

**YES**

requests an enhanced file format in a newly created SAS data file that counts observations beyond the 32-bit limitation. Although this SAS data file is created for an operating environment that stores the number of observations with a 32-bit integer, the data file behaves like a 64-bit file with respect to counters. This is the default.

**Restrictions**

A SAS data file that is created with an extended observation count is incompatible with releases prior to SAS 9.3. If the SAS data file was created in SAS 9.3 or later and EXTENDOBSCOUNTER was set to YES when the SAS data file was created, you must re-create the SAS data file with EXTENDOBSCOUNTER=NO.

**EXTENDOBSCOUNTER=**YES is valid only for an output SAS data file whose internal data representation stores the observation count as a 32-bit integer. EXTENDOBSCOUNTER=**YES is ignored for SAS data files with a 64-bit integer. For a table that lists the operating environments and the OUTREP= data representation values that are appropriate with EXTENDOBSCOUNTER=**YES, see “When Extending the Observation Count Is Supported” in SAS Language Reference: Concepts.

**NO**

specifies that the maximum observation count in a newly created SAS data file is determined by the long integer size for the operating environment. In operating environments with a 32-bit integer, the maximum number is 2^{31}–1 or approximately two billion observations (2,147,483,647). In operating environments with a 64-bit integer, the maximum number is 2^{63}–1 or approximately 9.2 quintillion observations.

See Also

“SAS Data File Encryption” in SAS Language Reference: Concepts
See Also

- “Extending the Observation Count for a 32-Bit SAS Data File” in SAS Language Reference: Concepts

Statements:

- “EXTENDOBSCOUNTER=YES | NO” in SAS Statements: Reference

System Options:

- “EXTENDOBSCOUNTER= System Option” in SAS System Options: Reference

FILECLOSE= Data Set Option

Specifies how a tape is positioned when a SAS data set is closed.

Valid in: DATA step and PROC steps

Category: Miscellaneous

See: FILECLOSE= Data Set Option in the documentation for the UNIX operating environment.

CAUTION: The option values are not recognized by all operating environments. Additional values are available on some operating environments. For more information about using SAS libraries that are stored on tape, see the SAS documentation for your operating environment.

Syntax

FILECLOSE= DISP | LEAVE | REREAD | REWIND

Syntax Description

DISP
positions the tape volume according to the disposition that is specified in the operating environment's control language.

LEAVE
positions the tape at the end of the file that was recently processed. Use FILECLOSE=LEAVE if you are not repeatedly accessing the same files in a SAS program, but you are accessing one or more subsequent SAS files on the same tape.

REREAD
positions the tape volume at the beginning of the file that was recently processed. Use FILECLOSE=REREAD if you are accessing the same SAS data set on tape several times in a SAS program.

REWIND
rewinds the tape volume to the beginning. Use FILECLOSE=REWIND if you are accessing one or more previous SAS files on the same tape, but you are not repeatedly accessing the same files in a SAS program.
FIRSTOBS= Data Set Option

Specifies the first observation that SAS processes in a SAS data set.

Valid in: DATA step and PROC steps
Category: Observation Control
Restrictions: Valid for input (read) processing only.

Cannot use with PROC SQL views.

Syntax

FIRSTOBS= n | nK | nM | nG | hexX | MIN | MAX

Syntax Description

n | nK | nM | nG
specifies the number of the first observation to process in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). For example, a value of 8 specifies the 8th observation, and a value of 3k specifies 3,072.

hexX
specifies the number of the first observation to process as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the 45th observation as the first observation to process.

MIN
sets the number of the first observation to process to 1. This is the default.

MAX
sets the number of the first observation to process to the maximum number of observations in the data set. This number can be up to the largest eight-byte, signed integer, which is $2^{63} - 1$, or approximately 9.2 quintillion observations.

Details

The FIRSTOBS= data set option affects a single, existing SAS data set. Use the FIRSTOBS= system option to affect all steps for the duration of your current SAS session.

FIRSTOBS= is valid for input (read) processing only. Specifying FIRSTOBS= is not valid for output or update processing.

You can apply FIRSTOBS= processing to WHERE processing. For more information, see “Processing a Segment of Data That Is Conditionally Selected” in SAS Language Reference: Concepts.

If you delete data set observations by using the VIEWTABLE window, the FIRSTOBS= option gives incorrect results if the value assigned to the FIRSTOBS= option is greater than the number of the observation that were deleted. This behavior occurs because deleting data set observations by using the VIEWTABLE window only flags the observation for deletion. The observation is not physically removed from the data set; it makes the observation unusable. For more information, see “Working with VIEWTABLE” in SAS Language Reference: Concepts.
Comparisons

- The FIRSTOBS= data set option overrides the FIRSTOBS= system option for the individual data set.
- When the FIRSTOBS= data set option specifies a starting point for processing, the OBS= data set option specifies an ending point. The two options are often used together to define a range of observations to be processed.
- When external files are read, the FIRSTOBS= option in the INFILE statement specifies which record to read first.

Example: Using the FIRSTOBS= Data Set Option

This PROC step prints the data set STUDY, beginning with observation 20:

```sas
proc print data=study(firstobs=20);
run;
```

This SET statement uses FIRSTOBS= and OBS= to read only observations 5 through 10 from the data set STUDY. The data set NEW contains six observations.

```sas
data new;
  set study(firstobs=5 obs=10);
run;
```

```sas
proc print data=new;
run;
```

See Also

Data Set Options:

- “OBS= Data Set Option” on page 41

Statements:

- “INFILE Statement” in SAS Statements: Reference
- “WHERE Statement” in SAS Statements: Reference

System Options:

- “FIRSTOBS= System Option” in SAS System Options: Reference

---

**GENMAX= Data Set Option**

Requests generations for a new data set, modifies the number of generations for an existing data set, and specifies the maximum number of versions.

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.
Syntax

GENMAX=number-of-generations

Syntax Description

number-of-generations requests generations for a data set and specifies the maximum number of versions to maintain. The value can be from 0 to 1,000. The default is GENMAX=0, which means that no generation data sets are requested.

Details

You use GENMAX= to request generations for a new data set and to modify the number of generations for an existing data set. The first time the data set is replaced, SAS keeps the replaced version and appends a four-character version number to its member name. The member name includes # and a three-digit number. For example, for a data set named A, a historical version would be A#001. After generations of a data set are requested, the member name is limited to 28 characters (rather than 32). The last four characters are reserved for the appended version number. When the GENMAX= data set option is set to 0, the member name can be up to 32 characters. If you reduce the number of generations for an existing data set, SAS deletes the oldest versions above the new limit.

Examples

Example 1: Requesting Generations When You Create a Data Set

The DATA step creates a data set named Work.A that can have as many as 10 generations (one current version and nine historical versions):

data a(genmax=10);
  x=1;
  output;
run;

Example 2: Modifying the Number of Generations on an Existing Data Set

The number of generations on the data set MYLIB.A is changed to 4:

proc datasets lib=mylib;
  modify a(genmax=4);
run;

See Also

- “Understanding Generation Data Sets” in SAS Language Reference: Concepts

Data Set Option:

- “GENNUM= Data Set Option” on page 31
GENNUM= Data Set Option

Specifies a particular generation of a SAS data set.

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with input data sets only.

**Syntax**

```
GENNUM=integer
```

**Syntax Description**

`integer` is a number that references a specific version from a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to the name of a data set. Specifying a negative number is a relative reference to a historical version in relation to the base (current) version, from the youngest to the oldest. Typically, a value of 0 refers to the base version.

The DATASETS procedure provides a variety of statements for which specifying GENNUM= has additional functionality:

- For the DATASETS and DELETE statements, GENNUM= supports the additional values ALL, HIST, and REVERT.
- For the DELETE procedure, GENNUM= supports the additional values ALL, HIST, and REVERT.
- For the CHANGE statement, GENNUM= supports the additional value ALL.
- For the CHANGE statement, specifying GENNUM=0 refers to all versions rather than the base version only.

**Details**

After generations for a data set have been requested using the GENMAX= data set option, use GENNUM= to request a specific version. For example, specifying GENNUM=3 refers to the historical version #003; specifying GENNUM=−1 refers to the youngest historical version.

After 999 replacements, the youngest version would be #999. After 1,000 replacements, SAS rolls over the youngest version number to #000. Therefore, if you want the historical version #000, specify GENNUM=1000.

Both an absolute reference and a relative reference refer to a specific version. A relative reference does not skip deleted versions. Therefore, when working with a generation group that includes one or more deleted versions, using a relative reference results in an error if the version that is being referenced has been deleted. For example, if you have the base version AIR and three historical versions (AIR#001, AIR#002, and AIR#003), delete AIR#002. The following statements return an error because AIR#002 does not exist.

```
proc print data=air (gennum= -2);
run;
```
Examples

**Example 1: Requesting a Version Using an Absolute Reference**
This example prints the historical version #003 for data set A, using an absolute reference:

```sas
proc print data=a(gennum=3);
run;
```

**Example 2: Requesting a Version Using a Relative Reference**
The following PRINT procedure prints the data set three versions back from the base version:

```sas
proc print data=a(gennum=-3);
run;
```

See Also
- “Understanding Generation Data Sets” in *SAS Language Reference: Concepts*
- “DATASETS” in *Base SAS Procedures Guide*

Data Set Option:
- “GENMAX= Data Set Option” on page 29

---

**IDXNAME= Data Set Option**
Directs SAS to use a specific index to match the conditions of a WHERE expression.

**Valid in:** DATA step and PROC steps  
**Category:** User Control of SAS Index Usage  
**Restrictions:**  
Use with input data sets only.  
Mutually exclusive with IDXWHERE= data set option

**Syntax**

`IDXNAME=index-name`

**Syntax Description**

`index-name` specifies the name (up to 32 characters) of a simple or composite index for the SAS data set. SAS does not attempt to determine whether the specified index is the best one or whether a sequential search might be more resource efficient.

**Interaction**  
The specification is not a permanent attribute of the data set and is valid only for the current use of the data set.

**Tip**  
To request that IDXNAME= usage be noted in the SAS log, specify the system option MSGLEVEL=I.
Details

To satisfy the conditions of a WHERE expression for an indexed SAS data set, SAS identifies zero or more candidate indexes that could be used to optimize the WHERE expression. From the list of candidate indexes, SAS determines the following:

- the candidate index that provides the best performance
- the rejection of all the indexes if a sequential pass of the data is more efficient

Because the index SAS selected cannot always provide the best optimization, you can direct SAS to use one of the candidate indexes by specifying the IDXNAME= data set option. If you specify an index that SAS does not identify as a candidate index, then IDXNAME= does not process the request. That is, IDXNAME= does not enable you to specify an index that would produce incorrect results.

Comparisons

IDXWHERE= enables you to override the SAS decision about whether to use an index, whereas INDEXNAME= enables you to direct SAS to use a specific index.

Example: Specifying an Index

This example uses the IDXNAME= data set option in order to direct SAS to use a specific index to optimize the WHERE expression. SAS then disregards the possibility that a sequential search of the data set might be more resource efficient. SAS does not attempt to determine whether the specified index is the best one. (The EMPNUM index was not created with the NOMISS option.)

```sas
data mydata.empnew;
  set mydata.employee (idxname=empnum);
  where empnum < 2000;
run;
```

See Also

- “Using an Index for WHERE Processing” in SAS Language Reference: Concepts

Data Set Option:

- “IDXWHERE= Data Set Option” on page 33

IDXWHERE= Data Set Option

Specifies whether SAS uses an index search or a sequential search to match the conditions of a WHERE expression.

- **Valid in:** DATA step and PROC steps
- **Category:** User Control of SAS Index Usage
- **Restrictions:** Use with input data sets only.
  - Mutually exclusive with IDXNAME= data set option

**Syntax**

`IDXWHERE= YES | NO`
Syntax Description
YES
 tells SAS to choose the best index to optimize a WHERE expression, and to disregard the possibility that a sequential search of the data set might be more resource efficient.

NO
 tells SAS to ignore all indexes and satisfy the conditions of a WHERE expression with a sequential search of the data set.

Note: You cannot use IDXWHERE= to override the use of an index to process a BY statement.

Details
By default, to satisfy the conditions of a WHERE expression for an indexed SAS data set, SAS decides whether to use an index or to read the data set sequentially. The software estimates the relative efficiency and chooses the method that is more efficient.

You might need to override the software's decision by specifying the IDXWHERE= data set option. The decision is based on general rules that might not always produce the best results. By specifying the IDXWHERE= data set option, you can determine the processing method.

Note: The specification is not a permanent attribute of the data set and is valid only for the current use of the data set.

Note: If you issue the system option MSGLEVEL=I, you can request that IDXWHERE= usage be noted in the SAS log if the setting affects index processing.

Comparisons
IDXNAME= enables you to direct SAS to use a specific index, whereas INDEXWHERE= enables you to override the SAS decision about whether to use an index.

Examples
Example 1: Specifying Index Usage
This example uses the IDXWHERE= data set option to tell SAS to decide which index is the best to optimize the WHERE expression. SAS then disregards the possibility that a sequential search of the data set might be more resource efficient:

```sas
data mydata.empnew;
  set mydata.employee (idxwhere=yes);
  where empnum < 2000;
```

Example 2: Specifying No Index Usage
This example uses the IDXWHERE= data set option to perform the following tasks:

- tell SAS to ignore any index
- satisfy the conditions of the WHERE expression with a sequential search of the data set

```sas
data mydata.empnew;
  set mydata.employee (idxwhere=no);
  where empnum < 2000;
```
INDEX= Data Set Option

Defines an index for a new output SAS data set.

Valid in: DATA step and PROC steps
Category: Data Set Control
Restriction: Use with output data sets only.

Syntax

INDEX= (index-specification <index-specification-2 ... > )

Syntax Description

index-specification
names and describes a simple or a composite index to be built. Index-specification has this form:

index= (variable(s) /UNIQUE /NOMISS )

index
is the name of a variable that forms the index or the name that you choose for a composite index.

variable(s)
is a list of variables to use in building a composite index.

UNIQUE
specifies that the values of the key variables must be unique. If you specify UNIQUE for a new data set and multiple observations have the same values for the index variables, the index is not built. A slash (/) must precede the UNIQUE option.

NOMISS
excludes all observations with missing values from the index. Observations with missing values are still read from the data set, but not through the index. A slash (/) must precede the NOMISS option.

Examples

Example 1: Defining a Simple Index
This INDEX= data set option defines a simple index for the SSN variable:

data new(index=(ssn));
Example 2: Defining a Composite Index
This INDEX= data set option defines a composite index named CITYST that uses the CITY and STATE variables:

data new(index=(cityst=(city state)));

Example 3: Defining a Simple and a Composite Index
This INDEX= data set option defines a simple index for SSN and a composite index for CITY and STATE:

data new(index=(ssn cityst=(city state)));

Example 4: Defining a Simple Index with the UNIQUE Option
This INDEX= data set option defines a simple index for the SSN variable with unique values:

data new(index=(ssn /unique));

Example 5: Defining a Simple Index with the NOMISS Option
This INDEX= data set option defines a simple index for the SSN variable, excluding all observations with missing values from the index:

data new(index=(ssn /nomiss));

Example 6: Defining Multiple Indexes By Using the UNIQUE and NOMISS Options
This INDEX= data set option defines a simple index for the SSN variable and a composite index for CITY and STATE. Each variable must have a UNIQUE and NOMISS option:

data new(index=(ssn /unique/nomiss cityst=(city state) /unique/nomiss));

See Also

• “INDEX CREATE Statement” in Base SAS Procedures Guide
• “CREATE INDEX Statement” in SAS SQL Procedure User’s Guide
• “Understanding SAS Indexes” in SAS Language Reference: Concepts

IN= Data Set Option

Creates a Boolean variable that indicates whether the data set contributed data to the current observation.

Valid in: DATA step
Category: Observation Control
Restriction: Use with the SET, MERGE, MODIFY, and UPDATE statements only.

Syntax

IN=variable
**Syntax Description**

`variable`

names the new variable whose value indicates whether the input data set contributed data to the current observation. Within the DATA step, the value of the variable is 1 if the data set contributed to the current observation. Otherwise, the value is 0.

**Details**

Specify the IN= data set option in parentheses after a SAS data set name in the SET, MERGE, MODIFY, and UPDATE statements only. Values of IN= variables are available to program statements during the DATA step. These variables are not included in the SAS data set that is being created, unless they are assigned to a new variable.

When you use IN= with BY-group processing, and when a data set contributes an observation for the current BY group, the IN= value is 1. The value remains as long as that BY group is still being processed and the value is not reset by programming logic.

**Example**

In this example, IN= creates a new variable, OVERSEAS, that denotes international flights. The variable `I` has a value of 1 when the observation is read from the NONUSA data set. Otherwise, the variable has a value of 0. The IF-THEN statement checks the value of `I` to determine whether the data set NONUSA contributed data to the current observation. If `I`=1, the variable OVERSEAS receives an asterisk (*) as a value.

```sas
data allflts;
  set usa nonusa(in=i);
  by fltnum;
  if i then overseas='*';
run;
```

**See Also**

- BY-GROUP Processing

**Statements:**

- “BY Statement” in *SAS Statements: Reference*
- “MERGE Statement” in *SAS Statements: Reference*
- “MODIFY Statement” in *SAS Statements: Reference*
- “SET Statement” in *SAS Statements: Reference*
- “UPDATE Statement” in *SAS Statements: Reference*

---

**KEEP= Data Set Option**

For an input data set, specifies the variables to process; for an output data set, specifies the variables to write to the data set.

**Valid in:** DATA step and PROC steps

**Category:** Variable Control
Syntax

\texttt{KEEP=variable(s)}

Syntax Description

\textit{variable(s)}

lists one or more variable names. You can list the variables in any form that SAS allows.

Details

If the \texttt{KEEP=} data set option is associated with an input data set, only those variables that are listed after the \texttt{KEEP=} data set option are available for processing. If the \texttt{KEEP=} data set option is associated with an output data set, only the variables that are listed after the option are written to the output data set. All variables are available for processing.

Comparisons

The \texttt{KEEP=} data set option differs from the \texttt{KEEP} statement in the following ways:

- In DATA steps, the \texttt{KEEP=} data set option can apply to input and output data sets. The \texttt{KEEP} statement applies only to output data sets.

- In DATA steps, when you create multiple output data sets, use the \texttt{KEEP=} data set option to write different variables to different data sets. The \texttt{KEEP} statement applies to all output data sets.

- In PROC steps, use only the \texttt{KEEP=} data set option, not the \texttt{KEEP} statement.

Example

In this example, only IDNUM and SALARY are read from PAYROLL, and they are the only variables in PAYROLL that are available for processing:

\begin{verbatim}
data bonus;
  set payroll(keep=idnum salary);
  bonus=salary*1.1;
run;
\end{verbatim}

See Also

Data Set Options:

- “\texttt{DROP=} Data Set Option” on page 17

Statements:

- “\texttt{KEEP Statement}” in \textit{SAS Statements: Reference}

\underline{LABEL= Data Set Option}

Specifies a label for a SAS data set.

\textbf{Valid in:} DATA step and PROC steps
Category: Data Set Control

Syntax

LABEL='label'

Syntax Description

'label'

specifies a text string of up to 256 characters. If the label text contains single quotation marks, enclose the label in double quotation marks. To remove a label from a data set, assign a blank space that is enclosed in quotation marks to the label.

You can also use two single quotation marks in the label text and enclose the string in single quotation marks.

Details

You can use the LABEL= option on input and output data sets. When you specify LABEL= on input data sets, it assigns a file label for the duration of that DATA step or PROC step. When you specify LABEL= for an output data set, the label becomes a permanent part of that file. The file can be printed using the CONTENTS or DATASETS procedure, and modified using PROC DATASETS.

A label that is assigned to a data set remains associated with that data set when you update a data set by using the APPEND procedure or the MODIFY statement. However, a label is lost if you use a data set with a previously assigned label to create a new data set in the DATA step. For example:

data one;
  set one;
run;

Comparisons

• The LABEL= data set option enables you to assign labels only for data sets. You can specify labels for the variables in a data set by using the LABEL statement.

• The LABEL= option enables you to assign labels to variables in the ATTRIB statement.

Example: Assigning Labels to Data Sets

data w2(label='1976 W2 Info, Hourly');
data new(label='Peter''s List');
data new(label="Hillside's Daily Account");
data sales(label='Sales For May(NE)');

See Also

Statements:

• “ATTRIB Statement” in SAS Statements: Reference

• “LABEL Statement” in SAS Statements: Reference
OBSBUF= Data Set Option

Determines the size of the view buffer for processing a DATA step view.

Valid in: DATA step and PROC steps
Category: Data Set Control
Restriction: Valid only for a DATA step view

Syntax

OBSBUF=\(n\)

Syntax Description

\(n\)
specifies the number of observations that are read into the view buffer at a time.

Default

32K bytes of memory are allocated for the default view buffer, which means the default number of observations that can be read into the view buffer at one time depends on the observation length. Therefore, the default is the number of observations that can fit into 32K bytes. If the observation length is larger than 32K bytes, then only one observation can be read into the buffer at a time.

Tip

To determine the observation length in bytes, use PROC CONTENTS for the DATA step view.

CAUTION

The maximum value for the OBSBUF= option depends on the amount of available memory. If you specify a value so large that the memory allocation of the view buffer fails, an out-of-memory error results.

Details

The OBSBUF= data set option specifies the number of observations that can be read into the view buffer at a time. The view buffer is a segment of memory that is allocated to hold output observations that are generated from a DATA step view. The size of the buffer determines how much data can be held in memory at one time. OBSBUF= enables you to tune the performance of reading data from a DATA step view.

The view buffer is shared between the request that opens the DATA step view (for example, a SAS procedure) and the DATA step view itself. Two computer tasks coordinate between requesting data and generating and returning the data as follows:

- When a request task (such as a PRINT procedure) requests data, task switching occurs from the request task to the view task. This action executes the DATA step
view and generates the observations. The DATA step view fills the view buffer with as many observations as possible.

- When the view buffer is full, task switching occurs from the view task back to the request task in order to return the requested data. The observations are cleared from the view buffer.

The size of the view buffer determines how many generated observations can be held. The number of generated observations determines how many times the computer must switch between the request task and the view task. For example, OBSBUF=1 results in task switching for each observation. OBSBUF=10 results in 10 observations being read into the view buffer at a time. The larger the view buffer, the less task switching is needed to process a DATA step view, which can speed execution time.

To improve efficiency, determine how many observations fit in the default buffer size, and then set the view buffer so that it can hold more generated observations.

Note: Using OBSBUF= can improve processing efficiency by reducing task switching. However, the larger the view buffer size, the more time it takes to fill. This process delays the task switching from the view task back to the request task. The delay is more apparent in interactive applications. For example, when you use the Viewtable window, the larger the view buffer, the longer it takes to display the requested observations. The view buffer must be filled before even one observation is returned to the Viewtable. Before you set a very large view buffer size, consider the following information:

- the type of application that you are using to process the DATA step view
- the amount of available memory

Example

For this example, the observation length is 10K. The default view buffer size, which is 32K, would result in three observations at a time to be read into the view buffer. The default view buffer size causes the execution time to be slower, because the computer must perform task switching for every three observations that are generated.

To improve performance, the OBSBUF= data set option is set to 100. This action causes 100 observations at a time to be read into the view buffer. It also reduces task switching in order to process the DATA step view with the PRINT procedure:

```sas
data testview / view=testview;
... more SAS statements ...
run;
proc print data=testview (obsbuf=100);
run;
```

See Also

Data Set Options:

- “SPILL= Data Set Option” on page 65
Valid in: DATA step and PROC steps
Category: Observation Control
Default: MAX
Restrictions: Use with input data sets only.
Cannot use with PROC SQL views

**Syntax**

\[ \text{OBS=} \ n \ | \ nK \ | \ nM \ | \ nG \ | \ nT \ | \ \text{hexX} \ | \ \text{MIN} \ | \ \text{MAX} \]

**Syntax Description**

- \( n \) \ | \ \( nK \) \ | \ \( nM \) \ | \ \( nG \) \ | \ \( nT \)
  
  specifies a number to indicate when to stop processing observations, with \( n \) as an integer. Using one of the letter notations results in multiplying the integer by a specific value. That is, specifying K (kilo) multiplies the integer by 1,024; M (mega) multiplies by 1,048,576; G (giga) multiplies by 1,073,741,824; or T (tera) multiplies by 1,099,511,627,776. For example, a value of 20 specifies 20 observations, and a value of 3m specifies 3,145,728 observations.

- \( \text{hexX} \)
  
  specifies a number to indicate when to stop processing as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the hexadecimal value F8 must be specified as 0F8X in order to specify the decimal equivalent of 248. The value 2DX specifies the decimal equivalent of 45.

- \( \text{MIN} \)
  
  specifies the number to indicate when to stop processing to 0. Use OBS=0 to create an empty data set that has the structure, but not the observations, of another data set.

**Interaction**

If OBS=0 and the NOREPLACE option is in effect, SAS can still take certain actions. SAS actually executes each DATA and PROC step in the program, using no observations. For example, SAS executes procedures, such as CONTENTS and DATASETS, that process libraries or SAS data sets.

- \( \text{MAX} \)
  
  specifies the number to indicate when to stop processing to the maximum number of observations in the data set. This number can be up to the largest 8-byte, signed integer, which is \(2^{63} - 1\), or approximately 9.2 quintillion. This is the default.

**Details**

OBS= tells SAS when to stop processing observations. To determine when to stop processing, SAS uses the value for OBS= in a formula that includes the value for OBS= and the value for FIRSTOBS=.

\[(\text{obs} - \text{firstobs}) + 1 = \text{results}\]

For example, if OBS=10 and FIRSTOBS=1 (which is the default for FIRSTOBS=), the result is 10 observations. That is, \((10 - 1) + 1 = 10\). If OBS=10 and FIRSTOBS=2, the result is nine observations. That is, \((10 - 2) + 1 = 9\). OBS= is valid only when an existing SAS data set is read.

The OBS= data set option overrides the OBS= system option for the individual data set.
Comparisons

- When the OBS= data set option specifies an ending point for processing, the FIRSTOBS= data set option specifies a starting point. The two options are often used together to define a range of observations to be processed.
- The OBS= data set option enables you to select observations from SAS data sets. You can select observations to be read from external data files by using the OBS= option in the INFILE statement.

Examples

**Example 1: Using OBS= to Specify When to Stop Processing Observations**

This example creates a SAS data set and executes the PRINT procedure with FIRSTOBS=2 and OBS=12. The result is 11 observations. That is, \((12 - 2) + 1 = 11\). The result of OBS= appears to be the observation number that SAS processes last.

```sas
data Ages;
  input Name $ Age;
datalines;
Miguel 53
Brad 27
Willie 69
Marc 50
Sylvia 40
Arun 25
Gary 40
Becky 51
Alma 39
Tom 62
Kris 66
Paul 60
Randy 43
Barbara 52
Virginia 72;
proc print data=Ages (firstobs=2 obs=12);
run;
```
Example 2: PROC PRINT Using a WHERE Statement

This example uses the data set that was created in Example 1, which contains 15 observations.

Here is the PRINT procedure with a WHERE statement. The subset of the data results in 12 observations:

```sas
proc print data=Ages;
  where Age LT 65;
run;
```
**Output 2.4**  PROC PRINT Output Using a WHERE Statement

The SAS System

<table>
<thead>
<tr>
<th>Obs</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Miguel</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>Brad</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>Marc</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>Sylvia</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>Arun</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>Gary</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>Becky</td>
<td>51</td>
</tr>
<tr>
<td>9</td>
<td>Alma</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>Tom</td>
<td>62</td>
</tr>
<tr>
<td>12</td>
<td>Paul</td>
<td>60</td>
</tr>
<tr>
<td>13</td>
<td>Randy</td>
<td>43</td>
</tr>
<tr>
<td>14</td>
<td>Barbara</td>
<td>52</td>
</tr>
</tbody>
</table>

**Example 3: PROC PRINT Using a WHERE Statement and OBS=**

Executing the PRINT procedure with the WHERE statement and OBS=10 results in 10 observations. That is, $(10 - 1) + 1 = 10$. With WHERE processing, SAS subsets the data and applies OBS= to the subset.

```sas
proc print data=Ages (obs=10);
   where Age LT 65;
run;
```
Output 2.5 PROC PRINT Output Using a WHERE Statement and OBS=

Example 4: PROC PRINT Using a WHERE Statement, OBS=, and FIRSTOBS=

The result of OBS= appears to be the observation number that SAS processes. If you apply FIRSTOBS=2 and OBS=10 to the subset, then the result is nine observations. That is, (10 - 2) + 1 = 9. OBS= is neither the observation number to end with nor how many observations to process; the value is used in the formula to determine when to stop processing.

```plaintext
proc print data=Ages (firstobs=2 obs=10);
  where Age LT 65;
run;
```
Example 5: PROC PRINT Showing Deleted Observations
This example uses the data set that was created in Example 1, with observation 6 deleted.

Here is PROC PRINT output of the modified file:

```sas
proc print data=Ages;
run;
```
**Example 6: PROC PRINT Using OBS=**

Executing the PRINT procedure with OBS=12 results in 12 observations. That is, \((12 - 1) + 1 = 12\):

```sas
proc print data=Ages (obs=12);
run;
```
Output 2.8  PROC PRINT Output Using OBS=

Example 7: Using OBS= When Observations Are Deleted

The result of OBS= appears to be the observation number that SAS processes. However, if you apply FIRSTOBS=2 and OBS=12, the result is 11 observations. That is, \((12 - 2) + 1 = 11\). OBS= is neither the observation number to end with nor how many observations to process; the value is used in the formula to determine when to stop processing.

```
proc print data=Ages (firstobs=2 obs=12);
run;
```
OUTREP= Data Set Option

Specifies the data representation for the output SAS data set.

Valid in: DATA step and PROC steps

Category: Data Set Control
See: OUTREP= data set option in the SAS Companion for z/OS.

Syntax

OUTREP= `format`

Syntax Description

`format` specifies the data representation, which is the form in which data is stored in a particular operating environment. Different operating environments use different standards or conventions for storing floating-point numbers (for example, IEEE or IBM mainframe); for character encoding (ASCII or EBCDIC); for the ordering of bytes in memory (big Endian or little Endian); for word alignment (4-byte boundaries or 8-byte boundaries); for integer data-type length (16-bit, 32-bit, or 64-bit); and for doubles (byte-swapped or not).

By default, SAS creates a new SAS data set by using the data representation of the CPU that is running SAS. Specifying the OUTREP= option enables you to create a SAS data set with a different data representation. For example, in a UNIX environment, you can create a SAS data set that uses a Windows data representation. For more information about compatibility and data representation, see “Processing Data Using Cross-Environment Data Access (CEDA)” in SAS Language Reference: Concepts.

Values for OUTREP= are listed in the following table:

<table>
<thead>
<tr>
<th>OUTREP= Value</th>
<th>Alias*</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA_TRU64</td>
<td>ALPHA_OSF</td>
<td>Tru64 UNIX</td>
</tr>
<tr>
<td>ALPHA_VMS_32</td>
<td>ALPHA_VMS</td>
<td>OpenVMS Alpha</td>
</tr>
<tr>
<td>ALPHA_VMS_64</td>
<td></td>
<td>OpenVMS Alpha</td>
</tr>
<tr>
<td>HP_IA64</td>
<td>HP_ITANIUM</td>
<td>HP-UX for the Itanium Processor Family Architecture</td>
</tr>
<tr>
<td>HP_UX_32</td>
<td>HP_UX</td>
<td>HP-UX for PA-RISC</td>
</tr>
<tr>
<td>HP_UX_64</td>
<td></td>
<td>HP-UX for PA-RISC, 64-bit</td>
</tr>
<tr>
<td>INTEL_ABI</td>
<td></td>
<td>ABI for Intel architecture</td>
</tr>
<tr>
<td>LINUX_32</td>
<td>LINUX</td>
<td>Linux for Intel architecture</td>
</tr>
<tr>
<td>LINUX_IA64</td>
<td></td>
<td>Linux for Itanium-based systems</td>
</tr>
<tr>
<td>LINUX_X86_64</td>
<td></td>
<td>Linux for x64</td>
</tr>
<tr>
<td>MIPS_ABI</td>
<td></td>
<td>MIPS ABI</td>
</tr>
<tr>
<td>OUTREP= Value</td>
<td>Alias*</td>
<td>Environment</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>MVS_32</td>
<td>MVS</td>
<td>31-bit SAS on z/OS</td>
</tr>
<tr>
<td>MVS_64_BFP</td>
<td></td>
<td>64-bit SAS on z/OS</td>
</tr>
<tr>
<td>OS2</td>
<td></td>
<td>OS/2 on Intel</td>
</tr>
<tr>
<td>RS_6000_AIX_32</td>
<td>RS_6000_AIX</td>
<td>AIX</td>
</tr>
<tr>
<td>RS_6000_AIX_64</td>
<td></td>
<td>AIX</td>
</tr>
<tr>
<td>SOLARIS_32</td>
<td>SOLARIS</td>
<td>Solaris for SPARC</td>
</tr>
<tr>
<td>SOLARIS_64</td>
<td></td>
<td>Solaris for SPARC</td>
</tr>
<tr>
<td>SOLARIS_X86_64</td>
<td></td>
<td>Solaris for x64</td>
</tr>
<tr>
<td>VAX_VMS</td>
<td></td>
<td>OpenVMS VAX</td>
</tr>
<tr>
<td>VMS_IA64</td>
<td></td>
<td>OpenVMS on HP Integrity</td>
</tr>
<tr>
<td>WINDOWS_32</td>
<td>WINDOWS</td>
<td>32-bit SAS on Microsoft Windows</td>
</tr>
<tr>
<td>WINDOWS_64</td>
<td></td>
<td>64-bit SAS on Microsoft Windows (for both Itanium-based systems and x64)</td>
</tr>
</tbody>
</table>

* It is recommended that you use the current values. The aliases are available for compatibility only.

**Details**

**CAUTION:**
Transcoding could result in character data loss when encodings are incompatible. For information about encoding and transcoding, see the *SAS National Language Support (NLS): Reference Guide*.

**See Also**
- “OUTREP=format” in *SAS Statements: Reference*
- “Processing Data Using Cross-Environment Data Access (CEDA)” in *SAS Language Reference: Concepts*

**POINTOBS= Data Set Option**

Specifies whether SAS creates compressed data sets whose observations can be randomly accessed or sequentially accessed.

**Valid in:** DATA step and PROC steps  
**Category:** Observation Control
Restriction: POINTOBS= is effective only when creating a compressed data set. Otherwise, it is ignored.

Syntax

POINTOBS= YES | NO

Syntax Description

YES
causes SAS software to produce a compressed data set that might be randomly accessed by observation number. This is the default.

Here are examples of accessing data directly by observation number:

• through the POINT= option of the MODIFY and SET statements in the DATA step
• through a specific observation number with PROC FSEDIT

Tips: Specifying POINTOBS= YES does not affect the efficiency of retrieving information from a data set. It does increase CPU usage by approximately 10% when creating a compressed data set and when updating or adding information to it.

NO
suppresses the ability to randomly access observations in a compressed data set by observation number.

Tips: If you do not need to access data by observation number in a compressed data set, then you can improve performance by approximately 10% when you specify POINTOBS= NO in these situations:

• when you create a compressed data set
• when you update or add observations to a compressed data set

Details

REUSE= YES takes precedence over POINTOBS= YES. For example:

data test (compress=yes pointobs=yes reuse=yes);

This data set option results in a data set that has POINTOBS= NO. Because POINTOBS= YES is the default when you use compression, REUSE= YES causes POINTOBS= to change to NO.

See Also

Data Set Options:

• “COMPRESS= Data Set Option” on page 13
• “REUSE= Data Set Option” on page 61

System Options:

• “COMPRESS= System Option” in SAS System Options: Reference
• “REUSE= System Option” in SAS System Options: Reference
**PW= Data Set Option**

Assigns a READ, WRITE, and ALTER password to a SAS file, and enables access to a password-protected SAS file.

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Note:** Check your log after this operation to ensure password security. For more information, see “Blotting Passwords and Encryption Key Values” in *SAS Language Reference: Concepts*.

**Syntax**

```
PW=password
```

**Syntax Description**

`password` must be a valid SAS name, which limits the password to eight characters and is case-insensitive. See “Words in the SAS Language” in *SAS Language Reference: Concepts*.

**Details**

The PW= option applies to all types of SAS files except catalogs. Use this option to assign a password to a SAS file or to access a password-protected SAS file.

When you replace a SAS data set that is protected by an ALTER password, the new data set inherits the ALTER password. To change the ALTER password for the new data set, use the MODIFY statement in the DATASETS procedure.

**Operating Environment Information**

For more information about using passwords, see the appropriate sections of the SAS documentation.

**Note:** A SAS password does not control access to a SAS file beyond the SAS system. Use the operating system-supplied utilities and file-system security controls to control access to SAS files outside of SAS.

**See Also**

- “File Protection” in *SAS Language Reference: Concepts*
- “Manipulating Passwords” in *Base SAS Procedures Guide*

**Data Set Options:**

- “ALTER= Data Set Option” on page 8
- “ENCRIPT= Data Set Option” on page 18
- “READ= Data Set Option” on page 55
- “WRITE= Data Set Option” on page 78
**PWREQ= Data Set Option**

Specifies whether to display a password dialog box.

- **Valid in:** DATA step and PROC steps
- **Category:** Data Set Control

**Syntax**

PWREQ= YES | NO

**Syntax Description**

YES

specifies to display a dialog box.

NO

prevents a dialog box from displaying. If a missing or invalid password is entered, the data set is not opened and an error message is written to the SAS log.

**Details**

In an interactive SAS session, the PWREQ= option controls whether a dialog box is displayed when an incorrect or a missing password for a password-protected SAS data set is specified. PWREQ= applies to data sets with READ=, WRITE=, or ALTER= passwords. PWREQ= is most useful in SCL applications.

**See Also**

Data Set Options:

- “ALTER= Data Set Option” on page 8
- “ENCRYPT= Data Set Option” on page 18
- “PW= Data Set Option” on page 54
- “READ= Data Set Option” on page 55
- “WRITE= Data Set Option” on page 78

---

**READ= Data Set Option**

Assigns a READ= password to a SAS file that prevents users from reading the file, unless they enter the password.

- **Valid in:** DATA step and PROC steps
- **Category:** Data Set Control
- **Note:** Check your log after this operation to ensure password security. For more information, see “Blotting Passwords and Encryption Key Values” in *SAS Language Reference: Concepts*. 
Syntax

READ= read-password

Syntax Description

read-password
must be a valid SAS name. For more information, see “Words in the SAS Language” in SAS Language Reference: Concepts.

Details

The READ= option applies to all types of SAS files except catalogs. Use this option to assign a password to a SAS file or to access a read-protected SAS file.

Note: A SAS password does not control access to a SAS file beyond the SAS system. Use the operating system-supplied utilities and file-system security controls to control access to SAS files outside of SAS.

See Also

• “File Protection” in SAS Language Reference: Concepts
• “Manipulating Passwords” in Base SAS Procedures Guide

Data Set Options:

• “ALTER= Data Set Option” on page 8
• “ENCRYPT= Data Set Option” on page 18
• “PW= Data Set Option” on page 54
• “WRITE= Data Set Option” on page 78

RENAME= Data Set Option

Changes the name of a variable.

Valid in: DATA step and PROC steps
Category: Variable Control

Syntax

RENAME=(old-variable-name-1=new-variable-name-1 <old-variable-name-2=new-variable-name-2 ...>)

Syntax Description

old-name
is the variable that you want to rename.

new-name
is the new name of the variable. It must be a valid SAS name.
Details

If you use the RENAME= data set option when you create a data set, the new variable name is included in the output data set. If you use RENAME= on an input data set, the new name is used in DATA step programming statements.

If you use RENAME= on an input data set that is used in a SAS procedure, SAS changes the name of the variable in that procedure. If you use RENAME= with WHERE processing such as a WHERE statement or a WHERE= data set option, the new name is applied before the data is processed. You must use the new name in the WHERE expression.

Use RENAME= in the same DATA step with either the DROP= data set option or the KEEP= data set option. The DROP= and KEEP= data set options are applied before RENAME=. You must use the old name in the DROP= and KEEP= data set options. You cannot drop and rename the same variable in the same statement.

Note: The RENAME= data set option has an effect only on data sets that are opened in output mode.

Use the RENAME statement or the RENAME= data set option when program logic requires that you rename variables. An example is two input data sets that have variables with the same name. To rename variables as a file management task, use the DATASETS procedure.

You must use the RENAME= data set option on the input data set or data sets to rename variables before processing begins.

Comparisons

The RENAME= data set option differs from the RENAME statement in these ways:

• You can use the RENAME= data set option, not the RENAME statement, in PROC steps.

• You must use the RENAME= data set option to rename different variables in different data sets. The RENAME statement applies to all output data sets.

Examples

Example 1: Renaming a Variable at Time of Output

This example uses RENAME= in the DATA statement to show that the variable is renamed when it is written to the output data set. The variable keeps its original name, X, during DATA step processing.

```
data one;
  input x y z;
data lines;
24 595 439
243 343 034
;proc print data=one;
run;

data two(rename=(x=keys));
  set one;
  z=x+y;
run;
proc print data=two;
```
Example 2: Renaming a Variable at Time of Input
This example renames variable X to a variable named KEYS in the SET statement, before DATA step processing.

```sas
data three;
  set one(rename=(x=keys));
  z=keys+y;
run;

proc print data=three;
run;
```

Output 2.11  Data Set Three

Example 3: Renaming a Variable for a SAS Procedure with WHERE Processing
This example renames variable `Score1` to a variable named `Score2` for the PRINT procedure. Because the new name is applied before the data is processed, the new name must be specified in the WHERE statement.
data test;
    input score1;
    datalines;
26
76
86
56
;

proc print data=test (rename=(score1=score2));
    where score2 gt 75;
run;

Output 2.12  Data Set Test

The SAS System

<table>
<thead>
<tr>
<th>Obs</th>
<th>score2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>76</td>
</tr>
<tr>
<td>3</td>
<td>86</td>
</tr>
</tbody>
</table>

See Also

Data Set Options:
- “DROP= Data Set Option” on page 17
- “KEEP= Data Set Option” on page 37

Statement:
- “RENAME Statement” in *SAS Statements: Reference*

Procedure:
- “DATASETS” in *Base SAS Procedures Guide*

---

**REPEMPTY= Data Set Option**

Specifies whether a new, empty data set can overwrite an existing SAS data set that has the same name.

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

**Syntax**

REPEMPTY=YES | NO
**Syntax Description**

**YES**

specifies that a new empty data set with a given name replaces an existing data set with the same name. This is the default.

**Interaction**

If REPEMPTY=YES and REPLACE=NO, then the data set is not replaced.

**NO**

specifies that a new empty data set with a given name does not replace an existing data set with the same name.

**Tips**

Use REPLACE=YES and REPEMPTY=NO for the convenience of replacing existing data sets with new ones that contain data.

Use REPEMPTY=NO to prevent the following syntax error from replacing the existing data set B with the new empty data set B that is created by mistake:

```
data mylib.a set b;
```

Use REPLACE=YES and REPEMPTY=NO for the protection of not overwriting existing data sets with new empty ones that are created by mistake.

**Details**

For an individual data set, the REPEMPTY= data set option overrides the REPEMPTY= option in the LIBNAME statement.

**Comparisons**

- The REPEMPTY= and REPLACE= data set options apply to permanent and temporary SAS data sets. However, the REPLACE system option only applies to permanent SAS data sets.

**See Also**

**Data Set Options:**

- “REPLACE= Data Set Option” on page 60

**Statement Options:**

- “REPEMPTY=YES|NO” in *SAS Statements: Reference*

**System Options:**

- “REPLACE System Option” in *SAS System Options: Reference*

---

**REPLACE= Data Set Option**

Specifies whether a new SAS data set that contains data can overwrite an existing data set that has the same name.
Valid in: DATA step and PROC steps
Category: Data Set Control
Restrictions: Use with output data sets only.
This option is valid only when creating a SAS data set.
Note: When you use the OUT2= PermanentLibrary._ALL_ option within PROC CONTENTS or PROC DATASETS with the CONTENTS statement, you must also set the REPLACE=YES data set option or the REPLACE system option.

Syntax

REPLACE=NO | YES

Syntax Description

NO
specifies that a new data set with a given name does not replace an existing data set with the same name.

YES
specifies that a new data set with a given name replaces an existing data set with the same name.

Details

• The REPLACE= data set option overrides the REPLACE system option for the individual data set.
• The REPLACE system option only applies to permanent SAS data sets.
Use REPLACE=YES and REPEMPTY=NO for the convenience of replacing existing data sets with new ones that contain data.

Example

Using the REPLACE= data set option in this DATA statement prevents SAS from replacing a permanent SAS data set named ONE in a library referenced by MYLIB:

data mylib.one(replace=no);

SAS writes a message to the log that the file has not been replaced.

See Also

System Options:
• “REPLACE System Option” in SAS System Options: Reference

REUSE= Data Set Option

Specifies whether new observations can be written to available space in compressed SAS data sets.

Valid in: DATA step and PROC steps
Category: Data Set Control
Restriction: Use with output data sets only.

Syntax

\texttt{REUSE=NO | YES}

Syntax Description

\textbf{NO}

does not track and reuse space in compressed data sets. New observations are appended to the existing data set. Specifying the \texttt{NO} argument results in less efficient data storage if you delete or update many observations in the SAS data set.

You can use procedures such as \texttt{APPEND} and \texttt{FSEDIT} that add observations to the end of SAS data sets with compressed data sets.

\textbf{YES}

tracks and reuses space in compressed SAS data sets. New observations are inserted in the space that is available when other observations are updated or deleted.

\texttt{REUSE=YES} causes new observations to be added wherever there is space in the file, not necessarily at the end of the file.

Details

By default, new observations are appended to existing compressed data sets. To track and reuse available space by deleting or updating other observations, use the \texttt{REUSE=} data set option when you create a compressed SAS data set.

Use \texttt{REUSE=} only when you are creating new data sets with the \texttt{COMPRESS= YES} data set option or system option.

The \texttt{REUSE=} data set option overrides the \texttt{REUSE=} system option.

\texttt{REUSE= YES} takes precedence over \texttt{POINTOBS= YES}. For example, the following statement results in a data set that has \texttt{POINTOBS= NO}:

\begin{verbatim}
data test(compress=yes pointobs=yes reuse=yes);
\end{verbatim}

Because \texttt{POINTOBS= YES} is the default when you use compression, \texttt{REUSE= YES} causes \texttt{POINTOBS=} to change to \texttt{NO}.

See Also

Data Set Options:

- “\texttt{COMPRESS=} Data Set Option” on page 13

System Options:

- “\texttt{REUSE=} System Option” in \textit{SAS System Options: Reference}

\textbf{ROLE= Data Set Option}

Identifies the fact table for a star schema join.

Valid in: PROC SQL
Syntax

\[ \text{ROLE} = \text{FACT} | \text{DIMENSION} | \text{DIM} \]

Syntax Description

FACT

identifies the SAS data set as the fact table for a star schema.

DIMENSION | DIM

identifies the SAS data set as a dimension table for a star schema.

Details

A star schema is an arrangement of several tables in which a large fact table is joined to several dimension tables. For example, you can join SAS data sets by using SQL procedure syntax to create a star schema.

To improve the performance of the application that processes the joined tables, specify the ROLE= data set option. For example, specify ROLE=FACT to designate the specific fact table. You can also specify ROLE=DIMENSION to designate each dimension table.

Because the role a table plays can change between queries, the ROLE= specification is in effect for the current step only and is not stored with the data set.

Example: Designating the Fact Table

In this example, the ROLE= data set option improves the performance of PROC SQL. ORDERS is the fact table, and PRODUCT, PERIOD, and CUSTOMER are dimension tables.

```
proc sql;
  select orders.Order_Total
  from orders (role=fact), product, period, customer
  where orders.Product_ID = product.Product_ID
    and orders.Period_ID = period.Period_ID
    and product.Product_Name = "camera"
    and period.Period_Name = "1997"
    and customer.Customer_Name = "Walmart";
quit;
```

See Also

“SQL” in SAS SQL Procedure User’s Guide
Syntax

\texttt{SORTEDBY=by-clause\slash collate-name | _NULL_}

Syntax Description

\texttt{by-clause < / collate-name} 
indicates how the data is currently sorted.

\texttt{by-clause}
names the variables and options that you use in a BY statement in a PROC SORT step.

\texttt{collate-name}
names the collating sequence that is used for the sort. By default, the collating sequence is that of your operating environment. A slash (/) must precede the collating sequence.

Operating Environment Information
For more information about collating sequences, see the SAS documentation for your operating environment.

\texttt{_NULL_}
removes any existing sort indicator.

Details

SAS determines whether a data set is already sorted by the key variable or variables in ascending order by checking the sort indicator. The sort indicator is stored in the data set descriptor information and is set from a previous sort. For more information about how the sort indicator is used and how it improves performance, see “The Sort Indicator” in \textit{SAS Language Reference: Concepts} and “SORTVALIDATE System Option” in \textit{SAS System Options: Reference}.

This example of the CONTENTS procedure indicates that the data set was sorted using the SORTEDBY= data set option.

\begin{verbatim}
Sort Information
Sortedby var1
Validated NO
Character Set ANSI
\end{verbatim}

Comparisons

\begin{itemize}
\item The CONTENTS statement in the DATASETS procedure indicates how a data set is sorted.
\item The SORTEDBY= option indicates how the data is sorted, but does not cause a data set to be sorted.
\end{itemize}

Example

This example uses the SORTEDBY= data set option to specify how the data is currently sorted. The data set ORDERS is sorted by PRIORITY and by the descending values of INDATE. Once the data set is created, the sort indicator is stored with it. These statements create the data set ORDERS and record the sort indicator:

\begin{verbatim}
64  Chapter 2 • Dictionary of Data Set Options

\end{verbatim}
libname mylib 'SAS-library';
options yearcutoff=1926;
data mylib.orders(sortedby=priority
descending indate);
input priority 1. +1 indate date7. 
   +1 office $ code $;
format indate date7.;
datalines;
1 03may01 CH J8U  
1 21mar01 LA M91  
1 01dec00 FW L6R  
1 27feb99 FW Q2A  
2 15jan08 FW I9U  
2 09jul99 CH P3Q  
3 08apr10 CH H5T  
3 31jan12 FW D2W 
;

Output 2.13  PROC CONTENTS Sort Information

<table>
<thead>
<tr>
<th>Sort Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sortedby</td>
</tr>
<tr>
<td>Validated</td>
</tr>
<tr>
<td>Character Set</td>
</tr>
</tbody>
</table>

See Also
- “DATASETS” in *Base SAS Procedures Guide*
- “SORT” in *Base SAS Procedures Guide*
- “SQL” in *SAS SQL Procedure User’s Guide*

---

**SPILL= Data Set Option**

Specifies whether to create a spill file for non-sequential processing of a DATA step view.

- **Valid in:** DATA step and PROC steps
- **Category:** Data Set Control
- **Restriction:** Valid only for a DATA step view

**Syntax**

`SPILL= YES | NO`

**Syntax Description**

- **YES**
  creates a spill file for non-sequential processing of a DATA step view. This is the default.
A spill file is never created for sequential processing of a DATA step view.

**NO** does not create a spill file or reduce the size of a spill file.

For direct (random) access, a spill file is always created even if you specify SPILL=NO.

If you do not have enough disk space to accommodate a resulting spill file from a DATA step view that generates a large amount of data, specify SPILL=NO.

For SAS procedures that process BY-group data, consider specifying SPILL=NO in order to write only the current BY group to the spill file.

When a DATA step view is opened for non-sequential processing, a spill file is created by default. The *spill file* contains the observations that are generated by a DATA step view. Subsequent requests for data read the observations from the spill file rather than execute the DATA step view again. The spill file is a temporary file in the Work library.

Non-sequential processing includes the following access methods, which are supported by several SAS statements and procedures. How the SPILL= data set option operates with each of the access methods is described here:

**random access**

retrieves observations directly either by an observation number or by the value of one or more variables through an index without reading all observations sequentially. Whether SPILL=YES or SPILL=NO, a spill file is always created, because the processing time to restart a DATA step view for each observation is significant.

**BY-group access**

uses a BY statement to process observations that are ordered, grouped, or indexed according to the values of one or more variables. SPILL=YES creates a spill file the size of all the data that is requested from the DATA step view. SPILL=NO writes only the current BY group to the spill file. The size of a spill file depends on the size of a BY group.

**two-pass access**

performs multiple sequential passes through the data. With SPILL=NO, no spill file is created. Instead, after the first pass through the data, the DATA step view is restarted for each subsequent pass through the data. If small amounts of data are returned by the DATA step view for each restart, the processing time to restart the view might be significant.

**Note:** With SPILL=NO, subsequent passes through the data could result in generating different data. Some processing might require using a spill file. For example, results from using random functions and computing values that are based on the current time of day could affect the data.
Examples

Example 1: Using a Spill File for a Small Number of Large BY Groups

This example creates a DATA step view that generates a large amount of random data and uses the UNIVARIATE procedure with a BY statement. The example illustrates the effects of SPILL= with a small number of large BY groups.

With SPILL=YES, all observations that are requested from the DATA step view are written to the spill file. With SPILL=NO, only the observations that are in the current BY group are written to the spill file. The output messages that are produced by this example show that the size of the spill file is reduced with SPILL=NO. However, the time it takes to truncate the spill file for each BY group might add to the overall processing time for the DATA step view.

```plaintext
options msglevel=i;
data vw_few_large / view=vw_few_large;
   drop i;
   do byval = 'Group A', 'Group B', 'Group C';
      do i = 1 to 500000;
         r = ranuni(4);
         output;
      end;
   end;
run;
proc univariate data=vw_few_large (spill=yes) noprint;
   var r;
   by byval;
run;
proc univariate data=vw_few_large (spill=no) noprint;
   var r;
   by byval;
run;
```
Example 2: Using a Spill File for a Large Number of Small BY Groups

This example creates a DATA step view that generates a large amount of random data and uses the UNIVARIATE procedure with a BY statement. This example illustrates the effects of SPILL= with a large number of small BY groups.

With SPILL=YES, all observations that are requested from the DATA step view are written to the spill file. With SPILL=NO, only the observations that are in the current BY group are written to the spill file. The output messages that are produced by this
example show that the size of the spill file is reduced with SPILL=NO. Small BY groups result in large space savings.

options msglevel=i;
data vw_many_small / view=vw_many_small;
  drop i;
  do byval = 1 to 100000;
    do i = 1 to 5;
      r = ranuni(4);
      output;
    end;
  end;
run;
proc univariate data=vw_many_small (spill=yes) noprint;
  var r;
  by byval;
run;
proc univariate data=vw_many_small (spill=no) noprint;
  var r;
  by byval;
run;
Log 2.2  SAS Log Output

options msglevel=i;
data vw_many_small / view=vw_many_small;
   drop i;

   do byval = 1 to 100000;
      do i = 1 to 5;
          r = ranuni(4);
          output;
      end;
   end;
run;

NOTE: DATA STEP view saved on file WORK.VW_MANY_SMALL.
NOTE: A stored DATA STEP view cannot run under a different operating system.
NOTE: DATA statement used (Total process time):
real time           0.56 seconds
    cpu time            0.03 seconds

proc univariate data=vw_many_small (spill=yes) noprint;
INFO: View WORK.VW_MANY_SMALL open mode: BY-group rewind.
   var r;
   by byval;
run;

INFO: View WORK.VW_MANY_SMALL opening spill file for output observations.
INFO: View WORK.VW_MANY_SMALL deleting spill file. File size was 8024240 bytes.
NOTE: View WORK.VW_MANY_SMALL.VIEW used (Total process time):
real time           30.73 seconds
    cpu time            29.59 seconds
NOTE: PROCEDURE UNIVARIATE used (Total process time):
real time           30.96 seconds
    cpu time            29.68 seconds

proc univariate data=vw_many_small (spill=no) noprint;
INFO: View WORK.VW_MANY_SMALL open mode: BY-group rewind.
   var r;
   by byval;
run;

INFO: View WORK.VW_MANY_SMALL opening spill file for output observations.
INFO: View WORK.VW_MANY_SMALL truncating spill file. File size was 65504 bytes.
NOTE: The above message was for the following by-group:
   byval=410
INFO: View WORK.VW_MANY_SMALL truncating spill file. File size was 65504 bytes.
NOTE: The above message was for the following by-group:
   byval=819
INFO: View WORK.VW_MANY_SMALL truncating spill file. File size was 65504 bytes.
NOTE: The above message was for the following by-group:
   byval=1229

INFO: View WORK.VW_MANY_SMALL truncating spill file. File size was 65504 bytes.
NOTE: The above message was for the following by-group:
   byval=99894
INFO: View WORK.VW_MANY_SMALL deleting spill file. File size was 32752 bytes.
NOTE: PROCEDURE UNIVARIATE used (Total process time):
real time           29.43 seconds
    cpu time            28.81 seconds

Example 3: Using a Spill File with Two-Pass Access
This example creates a DATA step view that generates a large amount of random data and uses the TRANSPOSE procedure. The example illustrates the effects of SPILL= with a procedure that requires two-pass access processing.

When PROC TRANSPOSE processes a DATA step view, the procedure must make two passes through the observations that the view generates. The first pass counts the number of observations and the second pass performs the transposition. With SPILL=YES, a spill file is created during the first pass, and the second pass reads the observations from the spill file. With SPILL=NO, a spill file is not created. After the first pass, the DATA step view is restarted.

The first TRANSPOSE procedure does not include the SPILL= data set option, even though a spill file is used by default. A SAS log message about the Open mode is not displayed.

```sas
options msglevel=i;
data vw_transpose/view=vw_transpose;
  drop i j;
  array x[10000];
  do i = 1 to 10;
    do j = 1 to dim(x);
      x[j] = ranuni(4);
    end;
  output;
  end;
run;
proc transpose data=vw_transpose out=transposed;
run;
proc transpose data=vw_transpose(spill=yes) out=transposed;
r
proc transpose data=vw_transpose(spill=no) out=transposed;
run;
```
Log 2.3  SAS Log Output

```
1   options msglevel=i;
2   data vw_transpose/view=vw_transpose;
3       drop i j;
4       array x[10000];
5       do i = 1 to 10;
6          do j = 1 to dim(x);
7             x[j] = ranuni(4);
8          end;
9         output;
10      end;
11   run;
NOTE: DATA STEP view saved on file WORK.VW_TRANSPOSE.
NOTE: A stored DATA STEP view cannot run under a different operating system.
NOTE: DATA statement used (Total process time):
   real time           0.68 seconds
   cpu time            0.18 seconds
12   proc transpose data=vw_transpose out=transposed;
13   run;
INFO: View WORK.VW_TRANSPOSE opening spill file for output observations.
INFO: View WORK.VW_TRANSPOSE deleting spill file.  File size was 880000 bytes.
NOTE: View WORK.VW_TRANSPOSE.VIEW used (Total process time):
   real time           2.37 seconds
   cpu time            1.17 seconds
NOTE: There were 10 observations read from the data set WORK.VW_TRANSPOSE.
NOTE: The data set WORK.TRANSPOSE has 10000 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
   real time           4.17 seconds
   cpu time            1.51 seconds
14   proc transpose data=vw_transpose (spill=yes) out=transposed;
INFO: View WORK.VW_TRANSPOSE open mode: sequential.
15   run;
INFO: View WORK.VW_TRANSPOSE reopen mode: two-pass.
INFO: View WORK.VW_TRANSPOSE opening spill file for output observations.
INFO: View WORK.VW_TRANSPOSE deleting spill file.  File size was 880000 bytes.
NOTE: View WORK.VW_TRANSPOSE.VIEW used (Total process time):
   real time           0.95 seconds
   cpu time            0.92 seconds
NOTE: There were 10 observations read from the data set WORK.VW_TRANSPOSE.
NOTE: The data set WORK.TRANSPOSE has 10000 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
   real time           1.34 seconds
   cpu time            1.32 seconds
16   proc transpose data=vw_transpose (spill=no) out=transposed;
INFO: View WORK.VW_TRANSPOSE open mode: sequential.
17   run;
INFO: View WORK.VW_TRANSPOSE reopen mode: two-pass.
INFO: View WORK.VW_TRANSPOSE restarting for another pass through the data.
NOTE: View WORK.VW_TRANSPOSE.VIEW used (Total process time):
   real time           1.42 seconds
   cpu time            1.40 seconds
NOTE: The View WORK.VW_TRANSPOSE was restarted 1 times. The following view
statistics
   only apply to the last view restart.
NOTE: There were 10 observations read from the data set WORK.VW_TRANSPOSE.
NOTE: The data set WORK.TRANSPOSE has 10000 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
   real time           1.01 seconds
   cpu time            0.98 seconds
```

See Also

Data Set Options:
**TOBSNO= Data Set Option**

Specifies the number of observations to send in a client/server transfer.

- **Valid in:** DATA step and PROC steps
- **Category:** Data Set Control
- **Restriction:** The TOBSNO= option is valid only for data sets that are accessed through a SAS server by using the REMOTE engine.

**Syntax**

```
TOBSNO=n
```

**Syntax Description**

- `n` specifies the number of observations to be transmitted.

**Details**

If the TOBSNO= option is not specified, its value is calculated based on the observation length and the size of the server’s transmission buffers. This action is specified by the PROC SERVER statement TBUFSIZE= option.

The TOBSNO= option is valid only for data sets that are accessed through a SAS server via the REMOTE engine. If this option is specified for a data set that is opened for update or accessed via another engine, it is ignored.

**See Also**

“FOPEN Function” in *SAS Functions and CALL Routines: Reference*
Details
Use the TYPE= data set option in a DATA step to perform these tasks:
• to create a special SAS data set in the proper format
• to identify the special type of SAS data set in a procedure statement
You can use the CONTENTS procedure to determine the type of data set.

Most SAS data sets do not have a specified type. However, there are several specially structured SAS data sets that are used by some SAS/STAT procedures. These SAS data sets contain special variables and observations, and they are usually created by SAS statistical procedures. Because most of the special SAS data sets are used with SAS/STAT software, they are described in the SAS/STAT User's Guide. Some of the special data sets are CORR, COV, SSPC, EST, and FACTOR.

Additional values are available in other SAS software products and are described in the appropriate documentation.

Note: If you use a DATA step with a SET statement to modify a special SAS data set, you must specify the TYPE= option in the DATA statement. The data-set-type is not automatically copied to the data set that is created.

See Also
“CONTENTS” in Base SAS Procedures Guide

WHERE= Data Set Option
Specifies specific conditions to use to select observations from a SAS data set.

Valid in: DATA step and PROC steps
Category: Observation Control
Restriction: Cannot be used with the POINT= option in the SET and MODIFY statements.

Syntax
WHERE=(where-expression-1 <logical-operator where-expression-2> )

Syntax Description
where-expression
is an arithmetic or logical expression that consists of a sequence of operators, operands, and SAS functions. An operand is a variable, a SAS function, or a constant. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. The expression must be enclosed in parentheses.

logical-operator
can be AND, AND NOT, OR, or OR NOT.

Details
• Use the WHERE= data set option with an input data set to select observations that meet the condition that is specified in the WHERE expression. SAS brings the observations into the DATA or PROC step for processing. Selecting observations that
meet the conditions of the WHERE expression is the first operation SAS performs in each iteration of the DATA step.

You can also select observations that are written to an output data set. In general, selecting observations at the point of input is more efficient than selecting them at the point of output. However, there are some cases when selecting observations at the point of input is not practical or not possible.

- You can apply OBS= and FIRSTOBS= processing to WHERE processing. For more information see “Processing a Segment of Data That Is Conditionally Selected” in *SAS Language Reference: Concepts*.

- You cannot use the WHERE= data set option with the POINT= option in the SET and MODIFY statements.

- You can use both the WHERE= data set option and the WHERE statement in the same DATA step. SAS ignores the WHERE statement for data sets with the WHERE= data set option. However, you can use the WHERE= data set option with the WHERE command in SAS/FSP software.

*Note:* Using indexed SAS data sets can improve performance significantly when you are using WHERE expressions to access a subset of the observations in a SAS data set. See “Understanding SAS Indexes” in *SAS Language Reference: Concepts* for a complete discussion of WHERE expression processing with indexed data sets and a list of guidelines to consider before indexing your SAS data sets.

**Comparisons**

- The WHERE statement applies to all input data sets, whereas the WHERE= data set option selects observations only from the data set for which it is specified.

- The DROP= and KEEP= data set options select variables for processing, whereas the WHERE= data set option selects observations.

**Examples**

**Example 1: Selecting Observations from an Input Data Set**

This example uses the WHERE= data set option to subset the SALES data set as it is read into another data set:

```sas
data whizmo;
  set sales(where=(product='whizmo'));
run;
```
Example 2: Selecting Observations from an Output Data Set

This example uses the WHERE= data set option to subset the SALES output data set:

```sas
data whizmo(where=(product='whizmo'));
  set sales;
run;
```

See Also

- “WHERE Statement” in *SAS Statements: Reference*
- WHERE-Expression Processing

WHEREUP= Data Set Option

Specifies whether to evaluate new observations and modified observations against a WHERE expression.

- **Valid in:** DATA step and PROC steps
- **Category:** Observation Control

**Syntax**

```
WHEREUP=NO | YES
```
Syntax Description

NO
   does not evaluate added observations and modified observations against a WHERE expression.

YES
   evaluates added observations and modified observations against a WHERE expression.

Details

Specify WHEREUP=YES when you want any added observations or modified observations to match a specified WHERE expression.

Examples

Example 1: Accepting Updates That Do Not Match the WHERE Expression

This example shows how WHEREUP= allows observations to be updated and added even though the modified observation does not match the WHERE expression:

```sas
data a;
x=1;
output;
x=2;
output;
run;
data a;
   modify a(where=(x=1) whereup=no);
x=3;
   replace; /* Update does not match WHERE expression */
   output; /* Add does not match WHERE expression */
run;
```

In this example, SAS updates the observation and adds the new observation to the data set.

Example 2: Rejecting Updates That Do Not Match the WHERE Expression

In this example, WHEREUP= does not allow observations to be updated or added when the update and the addition do not match the WHERE expression:

```sas
data a;
x=1;
output;
x=2;
output;
run;
data a;
   modify a(where=(x=1) whereup=yes);
x=3;
   replace; /* Update does not match WHERE expression */
   output; /* Add does not match WHERE expression */
run;
```
In this example, SAS does not update the observation nor does it add the new observation to the data set.

See Also

Data Set Option:

- “WHERE= Data Set Option” on page 74

WRITE= Data Set Option

Assigns a WRITE= password to a SAS file that prevents users from writing to a file, unless the users enter the password.

Valid in: DATA step and PROC steps

Category: Data Set Control

Note: Check your log after this operation to ensure password security. For more information, see “Blotting Passwords and Encryption Key Values” in SAS Language Reference: Concepts.

Syntax

WRITE= write-password

Syntax Description

write-password must be a valid SAS name. For more information, see “Words in the SAS Language” in SAS Language Reference: Concepts.

Details

The WRITE= option applies to all types of SAS files except catalogs. You can use this option to assign a password to a SAS file or to access a write-protected SAS file.

Note: A SAS password does not control access to a SAS file beyond the SAS system. You should use the operating system-supplied utilities and file-system security controls in order to control access to SAS files outside of SAS.

See Also

- “Manipulating Passwords” in Base SAS Procedures Guide

Data Set Options:

- “ALTER= Data Set Option” on page 8
- “ENCRYPT= Data Set Option” on page 18
- “PW= Data Set Option” on page 54
- “READ= Data Set Option” on page 55
Here is the recommended reading list for this title:

- *An Array of Challenges--Test Your SAS Skills*
- *Base SAS Glossary*
- *Base SAS Procedures Guide*
- *Combining and Modifying SAS Data Sets: Examples*
- *Learning SAS by Example*
- *SAS Formats and Informats: Reference*
- *SAS Functions and CALL Routines: Reference*
- *SAS Language Reference: Concepts*
- *SAS Scalable Performance Data Engine: Reference*
- *SAS Statements: Reference*
- *SAS System Options: Reference*
- *Step-by-Step Programming with Base SAS*
- *The Little SAS Book: A Primer*

For a complete list of SAS publications, go to [sas.com/store/books](http://sas.com/store/books). If you have questions about which titles you need, please contact a SAS Representative:

SAS Books  
SAS Campus Drive  
Cary, NC 27513-2414  
Phone: 1-800-727-0025  
Fax: 1-919-677-4444  
Email: sasbook@sas.com  
Web address: [sas.com/store/books](http://sas.com/store/books)
Index

A  
access methods  
  non-sequential processing 66
ALTER passwords  
  assigning to SAS files 54
ALTER= data set option 8

B  
buffers  
  allocated for data set processing 9
  page size and 11
  size of permanent buffer page 11
  view buffer size 40
BUFNO= data set option 9
BUFSIZE= data set option 11
BY-group access  
  spill files and 66

C  
client/server transfers  
  number of observations to send 73
CNTLLEV= data set option 12
COMPRESS= data set option 13
compressed data sets  
  reusing freed space 61
compressing data sets 13
  random vs. sequential access 52

D  
damaged data sets 16
data representation  
  output data sets 50
data set options 1
  examples 1
input data sets with 2
output data sets with 2
syntax 1
  system option interactions with 2
  data set types  
    for specially structured data sets 73
  data sets  
    buffers allocated for processing 9
    compressing 13
    conditions for selecting observations 74
    contributing to current observation 36
    damaged 16
dialog box for entering passwords 55
dropping variables 17
empty 59
encrypting 18
extension observation count 26
first observation to process in single data set 28
generations for 29
generations for, specifying 31
keeping variables 37
labels for 38
last observation for processing 41
overwriting 59, 60
repairing 16
replacing 60
selecting observations from 74
shared access level 12
sorting 63
specially structured 73
tape volume position when closing 27
updated, evaluating against WHERE expression 76
with same name 59, 60
DATA step views  
  spill file for non-sequential processing 65
  view buffer size 40
dialog boxes  
  for entering data set passwords 55
DLDMGACTION= data set option 16
DROP= data set option 17

E  
empty data sets 59
ENCRIPT= data set option 18
encryption  
  output data sets 18
ENCRYPTKEY data set option 23
EOC=
  EXTENDOBS roulette= data set option 26
  EXTENDOBSroulette= data set option 26

F
  fact tables
    for star schema joins 62
  FILECLOSE= data set option 27
  FIRSTOBS= data set option 28

G
  generations
    maximum number of versions 29
    modifying the number of 29
    requesting for data sets 29
    specifying for data sets 31
  GENMAX= data set option 29
  GENNUM= data set option 31

I
  IDXNAME= data set option 32
  IDXWHERE= data set option 33
  IN= data set option 36
  INDEX= data set option 35
  indexes
    damaged data sets and 16
    defining for output data sets 35
    overriding 33
    specifying a candidate index 32
    specifying index search 33
  input data sets
    data set options with 2
    excluding variables from processing 17
    selecting observations from 75
    specifying variables to process 37

J
  joins
    fact table for star schema join 62

K
  KEEP= data set option 37

L
  LABEL= data set option 38
  labels
    for data sets 38

N
  non-sequential processing
    access methods 66
    spill files and 65

O
  OBS= data set option 41
    comparisons 43
    data set with deleted observations 47
    WHERE processing with 44
  OBSBUF= data set option 40
    observation count
    extending 26
    observations
      client/server transfer 73
      compressing in output data set 13
      conditions for selecting 74
      contributing data sets for 36
      end point for processing 41
      extending count 26
      first observation to process in single data set 28
      updated data sets and WHERE expression 76
      writing to freed space 61
  output data sets
    compressing observations 13
    data representation 50
    data set options with 2
    defining indexes for 35
    encrypting 18
    excluding variables from being written 17
    extending observation count 26
    selecting observations from 76
    size of permanent buffer page 11
    specifying variables to write 37
  OUTREP= data set option 50

P
  page size
    buffers and 11
  password-protected files
    enabling access to 54
  passwords
    ALTER passwords 8
    assigning to SAS files 54
    dialog box for entering 55
    READ passwords 55
    WRITE passwords 78
  permanent buffer page
    size of 11
  POINTOBS= data set option 52
  procedures
WHERE processing with, renaming variables 58
PW= data set option 54
PWREQ= data set option 55

R
random access
spill files and 66
RDC (Ross Data Compression) 13
READ passwords
assigning to SAS files 54, 55
READ= data set option 55
RENAME= data set option 56
renaming variables 56
at time of input 58
at time of output 57
for procedures with WHERE processing 58
repairing data sets 16
REPEMPTY= data set option 59
REPLACE= data set option 60
REUSE= data set option 61
RLE (Run Length Encoding) 13
ROLE= data set option 62
Ross Data Compression (RDC) 13
Run Length Encoding (RLE) 13

S
SAS files
ALTER passwords 8
assigning passwords to 54
preventing reading 55
preventing writing to 78
shared access level
for data sets 12
sort information
for data sets 63
SORTEDBY= data set option 63
sorting
data set sort information 63
spill files 65
SPILL= data set option 65
star schema joins
fact table for 62
system options
data set interactions with 2

T
tape volume
position when closing data set 27
TOBSNO= data set option 73
two-pass access
spill files and 66
TYPE= data set option 73

V
variables
dropping from data set processing 17
keeping for data set processing 37
renaming 56
view buffers 40
size of 40

W
WHERE expressions
evaluating updated data sets against 76
index search for 33
overriding indexes 33
sequential search for 33
specifying an index to match conditions 32
WHERE processing
OBS= data set option with 44
renaming variables in procedures 58
WHERE= data set option 74
WHEREUP= data set option 76
WRITE passwords
assigning to SAS files 54, 78
WRITE= data set option 78
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, in the U.S. and other countries. © 2015 SAS Institute Inc. All rights reserved.