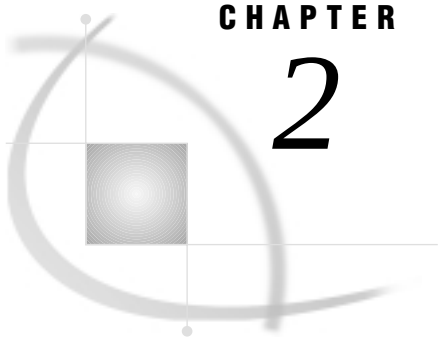


CHAPTER

2 ACCESS Procedure Reference

Introduction

You use the ACCESS procedure to create descriptor files that enable SAS/ACCESS software to access supported PC file formats. This chapter provides general reference information for the ACCESS procedure; more detailed information specific to supported file formats is provided in Chapters 5 and later.

In this chapter, the PROC ACCESS statement and its options are presented first, followed by the procedure statements. The last section, “Performance and Efficient View Descriptors” on page 31, presents several efficiency considerations for using SAS/ACCESS software.

For a quick reference to ACCESS procedure syntax, see the tab page enclosed with your PC file format chapter.

For full information, refer to your base SAS reference documentation and the SAS documentation for your operating environment.

To get online help for the ACCESS procedure, select the **Help** menu.

Note to UNIX and OS/390 Users

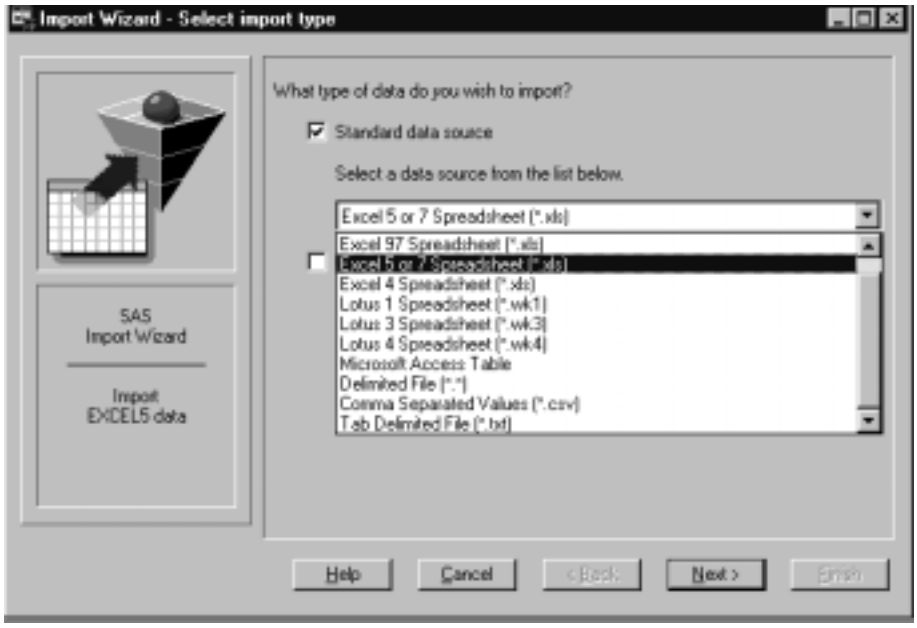
If you are running this SAS/ACCESS interface under the UNIX or OS/390 operating environment, this chapter does not apply to you. Instead, see Chapter 3, “DBF and DIF Procedures,” on page 33. Under UNIX and PC hosts, you can use these procedures to convert a DBF or DIF file to a SAS data set or a SAS data set to a DBF or DIF file. Under OS/390, you can use PROC DBF *only* to convert a DBF file to a SAS data set or a SAS data set to a DBF file.

Import/Export Facility

UNIX and PC users can access DBF, WK*n*, Excel 4 and 5, Excel 97, and MS Access and other data through the Import/Export facility or by using the IMPORT and EXPORT procedures. An overview is included in Chapter 5, "Import/Export Facility and Procedures," on page 51.

To use the point-and-click interface from a SAS PROGRAM EDITOR window, select the **File** menu and then select the **Import Data** or **Export Data** item. Information about how to import or export PC file data is available from the Help button. The following is a sample Import window:

Display 2.1 Import Window



To write code to import or export PC file data, refer to the detailed descriptions of the IMPORT and EXPORT procedures in the *SAS Procedures Guide*. This documentation also includes several examples.

ACCESS Procedure Syntax

PROC ACCESS *options*;

Create and Update Statements

CREATE *libref.member-name*.ACCESS|VIEW ;

UPDATE *libref.member-name*.ACCESS|VIEW ;

Database-Description Statement

PATH= *'path-and-filename<.PC-filename-extension>'* | <'>*filename*<'> | *fileref*;

Editing Statements

ASSIGN<=>YES|NO|Y|N;

DROP <'>*column-identifier-1*<'>

<<'>...*column-identifier-n*<'>>;

FORMAT <'>*column-identifier-1*<'><=>*SAS-format-name-1*<'>
<...<'>*column-identifier-n*<'><=>*SAS-format-name-n*>;

LIST <ALL|VIEW|<'>*column-identifier*<'>>;

QUIT;

RENAME <'>*column-identifier-1*<'><=>*SAS-variable-name-1*
<...<'>*column-identifier-n*<'><=>*SAS-variable-name-n*>;

RESET ALL | <'>*column-identifier-1* <'><...<'>*column-identifier-n*<'>>;

SELECT ALL|<'>*column-identifier-1*<'>
<...<'>*column-identifier-n*<'>>;

SUBSET *selection-criteria*;

UNIQUE<=>YES|NO|Y|N ;

RUN;

The file-specific statements for your PC files might differ from those listed above. See your PC file format chapter for your file's statements.

Description

Use the ACCESS procedure to create access descriptors, view descriptors, and SAS data files. Descriptor files describe PC file data so that you can directly read, update, or extract the PC file data from within a SAS program. The following sections explain the statements and options that might appear in a PROC ACCESS procedure.

PROC ACCESS Statement Options

The syntax of the ACCESS procedure statement is

PROC ACCESS *options*;

The PROC ACCESS statement options available with all supported PC file formats are described below. Other options, specific to particular PC file formats, are described in the DBMS-specific chapters.

DBMS=*pc-file-format*
specifies which PC database product or spreadsheet system you want to access from SAS. This is the only required option. The valid types are DBF, DIF, WK1, WK3, WK4, and XLS.

ACCDESC=*libref.access-descriptor* <(READ|WRITE|ALTER=*password*)>
specifies an existing access descriptor.

Use this option when creating or updating a view descriptor based on an access descriptor that was created in a separate PROC ACCESS step.

You name the view descriptor in the CREATE statement. You can also use a SAS data set option on the ACCDESC= option to specify a SAS password for the access descriptor.

The ACCDESC= option has two aliases: AD= and ACCESS=.

VIEWDESC=*libref.view-descriptor*
specifies a view descriptor as input for the OUT= option. (See the description of OUT=.)

OUT= *<libref.>member-name*
specifies a SAS data file. When VIEWDESC= and OUT= are used together, you can write data accessed from the view descriptor to the SAS data set specified in OUT=. For example:

```
proc access viewdesc=vlib.invq4
            out=dlib.invq4;
run;
```

CAUTION:
Altering a PC file might invalidate defined descriptors. Altering the format of a PC file that has descriptor files defined on it might cause these descriptors to be out-of-date or invalid. For example, if you add a column to a file and an existing access descriptor is defined on that file, the access descriptor and any view descriptors based on it do not show the new column. You must re-create the descriptors to be able to show and select the new column. △

SAS System Passwords for SAS/ACCESS Descriptors

The SAS System enables you to control access to SAS data sets and access descriptors by associating one or more SAS System passwords with them. You must first create the descriptor files before assigning SAS passwords to them, as described in "Assigning Passwords" on page 15. Table 2.1 on page 14 summarizes the levels of protection that SAS System passwords have and their effects on access descriptors and view descriptors.

Table 2.1 Password and Descriptor Interaction

	READ=	WRITE=	ALTER=
access descriptor	no effect on descriptor	no effect on descriptor	protects descriptor from being read or edited
view descriptor	protects PC file data from being read or updated	protects PC file data from being updated	protects descriptor from being read or edited

When you create view descriptors, you can use a SAS data set option after the ACCDESC= option to specify the access descriptor's password (if one exists). In this case, you are *not* assigning a password to the view descriptor that is being created. Rather, using the password grants you permission to use the access descriptor to create the view descriptor. For example:

```
proc access dbms=dbf
     accdesc=adlib.customer(alter=rouge);
  create vlib.customer.view;
  select all;
run;
```

By specifying the ALTER-level password, you can read the ADLIB.CUSTOMER access descriptor and therefore create the VLIB.CUSTOMER view descriptor.

For detailed information on the levels of protection and the types of passwords you can use, refer to your base SAS software documentation.

Assigning Passwords

To assign, change, or delete a SAS password, use the DATASETS procedure's MODIFY statement. Here is the basic syntax for using PROC DATASETS to assign a password to an access descriptor, a view descriptor, or a SAS data file:

PROC DATASETS LIBRARY= *libref* MEMTYPE= *member-type*;
MODIFY *member-name* (*password-level= password-modification*);
RUN;

In this syntax statement, the *password-level* argument can have one or more of the following values: READ=, WRITE=, ALTER=, or PW=. PW= assigns read, write, and alter privileges to a descriptor or data file. The *password-modification* argument enables you to assign a new password or to change or delete an existing password. For example, this PROC DATASETS statement assigns the password MONEY with the ALTER level of protection to the access descriptor ADLIB.SALARIES

```
proc datasets library=adlib memtype=access;
   modify salaries (alter=money);
run;
```

In this case, you are prompted for the password whenever you try to browse or edit the access descriptor or to create view descriptors that are based on ADLIB.SALARIES.

In the next example, the PROC DATASETS statement assigns the passwords MYPW and MYDEPT with READ and ALTER levels of protection to the view descriptor VLIB.JOBC204:

```
proc datasets library=vlib memtype=view;
   modify jobc204 (read=mypw alter=mydept);
run;
```

In this case, you are prompted for the SAS passwords when you try to read the PC file data, or try to browse or edit the view descriptor VLIB.JOBC204 itself. You need both levels to protect the data and descriptor from being read. However, you could still update the data accessed by VLIB.JOBC204, for example, by using a PROC SQL UPDATE statement. Assign a WRITE level of protection to prevent data updates.

To delete a password on an access descriptor or any SAS data set, put a slash after the password:

```
proc datasets library=vlib memtype=view;
   modify jobc204 (read=mypw/ alter=mydept/);
run;
```

Procedure Statements

This section describes in alphabetical order the statements you use inside a PROC ACCESS program block to create or modify access and view descriptors. Table 2.2 on page 16 presents a task-oriented overview and indicates the order in which statements must appear. See “CREATE” on page 17 for additional information on this order and for information on database-description and editing statements.

Table 2.2 Options and Statements Required for the ACCESS Procedure

Tasks	Options and Statements You Use
create an access descriptor	**PROC ACCESS** DBMS=DBF|DIF|WK*n*|XLS; **CREATE** *libref.member-name*.ACCESS; *required database-description statements*; *optional editing statements*; **RUN**;
create an access descriptor and a view descriptor	**PROC ACCESS** DBMS=DBF|DIF|WK*n*| XLS; **CREATE** *libref.member-name*.ACCESS; *required database-description statements*; *optional editing statements*; **CREATE** *libref.member-name*.VIEW; **SELECT** *column-list*; *optional editing statements*; **RUN**;
create a view descriptor from an existing access descriptor	**PROC ACCESS** DBMS=DBF|DIF|WK*n*|XLS ACCDESC=*libref.access-descriptor*; **CREATE** *libref.member-name*.VIEW; **SELECT** *column-list*; *optional editing statements*; **RUN**;

As the table indicates, you can create one or more access descriptors and view descriptors in one execution of PROC ACCESS, or you can create the descriptors in separate executions.

Dictionary

ASSIGN

Indicates whether SAS variable names and formats are automatically generated.

Optional statement
Applies to: access descriptor
Interacts with: FORMAT, RENAME, RESET, UNIQUE
Not allowed with: UPDATE
Default: NO

Syntax

ASSIGN <=> YES|NO|Y|N;

Details The ASSIGN statement indicates whether SAS variable names and formats are automatically generated. Where long names must be shortened to the SAS length limit of 8 characters, variable names are automatically generated.

An editing statement such as ASSIGN appears after the CREATE and database-description statements. See "CREATE" on page 17 for more information.

The value NO (or N) enables you to modify SAS variable names and formats when you create an access descriptor and when you create view descriptors that are based on this access descriptor. During an access descriptor's creation, you use the RENAME statement to change SAS variable names; you use the FORMAT statement to change SAS formats.

Specify a YES (or Y) value for this statement to generate unique SAS variable names from the first 8 characters of the PC file column names, according to the rules listed below. With YES, you can change the SAS variable names only in the access descriptor. The SAS variable names that are saved in an access descriptor are *always* used when view descriptors are created from the access descriptor; you cannot change them in the view descriptors.

Default SAS variable names are generated according to these rules:

- If the column name is longer than 8 characters, the SAS System uses only the first 8 characters. If truncating results in duplicate names, numbers are appended to the ends of the names to prevent duplicate names. For example, the names clientsname and clientsnumber become the SAS names clientsn and clients0.
- If the column name in the PC file contains blank characters, the SAS System ignores the blank characters. For example, the column name PAID ON becomes the SAS name PAIDON.
- If the column name in the PC file starts with a digit (0 through 9), the SAS System adds the character Z before it. For example, the column name 1STYEAR becomes the SAS name Z1STYEAR.
- If the column name contains characters that are invalid in SAS names (including national characters), the SAS System replaces the invalid characters with underscores (_). For example, the column name $PAID becomes the SAS variable name _PAID.

If you specify YES for this statement, the SAS System automatically resolves any duplicate variable names. However, if you specify YES, you cannot specify the RENAME, FORMAT, RESET, or UNIQUE statements when you create view descriptors that are based on the access descriptor. When you are updating an access descriptor, you cannot specify the ASSIGN statement.

When the SAS/ACCESS interface encounters the next CREATE statement to create an access descriptor, the ASSIGN statement is reset to the default NO value.

AN is the alias for the ASSIGN statement.

CREATE

Creates a SAS/ACCESS descriptor file.

Required statement

Applies to: access descriptor or view descriptor

Syntax

CREATE *libref.descriptor-name*.ACCESS | VIEW;

Details Use CREATE to create an access or view descriptor for a PC file you want to access from the SAS system. To access a particular PC file of a supported type, you must create first an access descriptor, and then one or more view descriptors based on the access descriptor.

The descriptor name has three parts, separated by periods(.). The *libref* identifies a SAS data library, which is associated with a directory on the local system's disk where the descriptor will be created. The *libref* must already have been created using the LIBNAME statement. The *descriptor-name* is the name of the descriptor to be created. The third part is the descriptor type. Specify ACCESS for an access descriptor or VIEW for a view descriptor.

You can use the CREATE statement as many times as necessary in one procedure execution. That is, you can create multiple access descriptors, as well as one or more view descriptors based on these access descriptors, within the same execution of the ACCESS procedure. Or, you can create access descriptors and view descriptors in separate executions of the procedure.

You can use CREATE and UPDATE in the same PROC ACCESS block with one restriction: a CREATE statement for a view descriptor may not follow an UPDATE statement.

Creating access descriptors

When you create an access descriptor, you must place statements or groups of statements in a certain order after the PROC ACCESS statement and its options, as listed here:

1. CREATE must be the first statement after the PROC ACCESS statement with one exception: if the block includes both CREATE and UPDATE statements, either statement may be the first in the block.
2. Next, specify any database-description statement, such as PATH=. This information describes the location and characteristics of the PC file. These statements must be placed before any editing statements. Do not specify these statements when you create view descriptors.

   Information from database-description statements is stored in an access descriptor. Therefore, you do not repeat this information when you create view descriptors. See Chapters 5 and later for additional database-description statements for your PC file format.
3. Next, specify any editing statements: ASSIGN, DROP, FORMAT, LIST, RENAME, RESET, and SUBSET. QUIT is also an editing statement, but using it terminates PROC ACCESS without creating your descriptor.
4. Finally, specify the RUN statement. RUN executes the ACCESS procedure.

The order of the statements within the database-description and editing groups sometimes matters; see the individual statement descriptions for more information.

Note: Altering a PC file that has descriptor files defined on it may cause the descriptor files to be out-of-date or invalid. For example, if you re-create a file and add a new column to the file, an existing access descriptor defined on that file does not show that column; in this case, the descriptor may still be valid. However, if you re-create a

file and delete an existing column from the file, the descriptor may be invalid. If the deleted column is included in a view descriptor and this view is used in a SAS program, the program fails and an error message is written to the SAS log. △

Creating view descriptors

You can create view descriptors and access descriptors in the same ACCESS procedure or in separate procedures.

To create a view descriptor and the access descriptor on which it is based within the *same* PROC ACCESS execution, you must place the statements or groups of statements in a particular order after the PROC ACCESS statement and its options, as listed below:

1. First, create the access descriptor as described in "Creating access descriptors" on page 18 except omit the RUN statement.
2. Next, specify the CREATE statement for the view descriptor. The CREATE statement must follow the PROC ACCESS statements that you used to create the access descriptor.
3. Next, specify any editing statements: SELECT, SUBSET, and UNIQUE are valid only when creating view descriptors. FORMAT, LIST, RENAME, and RESET are valid for both view and access descriptors. FORMAT, RENAME, and UNIQUE can be specified only when ASSIGN=NO is specified in the access descriptor referenced by this view descriptor. QUIT is also an editing statement but using it terminates PROC ACCESS without creating your descriptor.

   The order of the statements within this group usually does not matter; see the individual statement descriptions for any restrictions.
4. Finally specify the RUN statement. RUN executes PROC ACCESS.

To create a view descriptor based on an access descriptor that was created in a *separate* PROC ACCESS step, you specify the access descriptor's name in the ACCDESC= option in the new PROC ACCESS statement. You must specify the CREATE statement before any of the editing statements for the view descriptor.

If you create only one descriptor in a PROC step, the CREATE statement and its accompanying statements are checked for errors when you submit PROC ACCESS for processing. If you create multiple descriptors in the same PROC step, each CREATE statement (and its accompanying statements) is checked for errors as it is processed.

If no errors are found when the RUN statement is processed, all descriptors are saved. If errors are found, error messages are written to the SAS log, and processing is terminated. After you correct the errors, resubmit your statements.

Examples

The following example creates the access descriptor ADLIB.PRODUCT for the worksheet file named **c:\sasdemo\specprod.wk4**:

```
libname adlib 'c:\sasdata';

proc access dbms=wk4;
   create adlib.product.access;
   path='c:\sasdemo\specprod.wk4';
   getnames=yes;
   assign=yes;
   rename productid prodid
          fibername fiber;
   format productid  4.
          weight     e16.9
          fibersize  e20.13
```

```
          width      e16.9;
run;
```

The following example creates an access descriptor named ADLIB.EMPLOY for the Excel worksheet named **c:\dubois\employ.xls**. It also creates a view descriptor named VLIB.EMP1204 for this same file:

```
libname adlib 'c:\sasdata';
libname vlib 'c:\sasviews';

proc access dbms=xls;
   /* create access descriptor  */
   create adlib.employ.access;
   path='c:\dubois\employ.xls';
   getnames=yes;
   assign=no;
   list all;

   create vlib.emp1204.view;
   /* create view descriptor  */
   select empid lastname hiredate salary
          dept gender birthdate;
   format empid 6.
          salary dollar12.2
          jobcode 5.
          hiredate datetime7.
          birthdate datetime7.;
   subset where jobcode=1204;
run;
```

The following example creates a view descriptor VLIB.BDAYS from the ADLIB.EMPLOY access descriptor, which was created in the previous PROC ACCESS step. Note that FORMAT could be used because the access descriptor was created with ASSIGN=NO.

```
libname adlib 'c:\sasdata';
libname vlib 'c:\sasviews';

proc access accdesc=adlib.employ;
   create vlib.bdays.view;
   select empid lastname birthdate;
   format empid 6.
          birthdate datetime7.;
run;
```

DROP

Drops a column from a descriptor.

Optional statement

Applies to: access descriptor, view descriptor

Interacts with: RESET, SELECT, UPDATE

Syntax

DROP <'>*column-identifier-1*<'>
<...<'>*column-identifier-n*<'>>;

Details

The DROP statement drops the specified column from an access descriptor. The column cannot be selected for a view descriptor that is based on the access descriptor. However, the specified column in the PC file remains unaffected by this statement.

Note: The DROP statement can only be specified when creating or updating an access descriptor, or when you are updating a view descriptor. DROP is not allowed when you are creating a view descriptor. When you use the UPDATE statement, you can specify DROP to remove a column from the view descriptor. However, the specified column in the PC file remains unaffected by the DROP statement. △

An editing statement, such as DROP, must follow the CREATE and database-description statements when you create an access descriptor. See "CREATE" on page 17 for more information on the order of statements.

The *column-identifier* argument can be either the column name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the access descriptor or view descriptor. For example, to drop the third and fifth columns, submit the following statement:

```
drop 3 5;
```

If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotes. You can drop as many columns as you want in one DROP statement.

To display a column that was previously dropped, specify that column name in the RESET statement. However, doing so also resets all the column's attributes (such as SAS variable name, format, and so on) to their default values.

FORMAT

Changes a SAS format for a PC file column.

Optional statement

Applies to: access descriptor or view descriptor

Interacts with: ASSIGN, DROP, RESET

Syntax

FORMAT|FMT <'>*column-identifier-1*<'><=>
SAS-format-name-1
<...<'>*column-identifier-n*<'><=>
SAS-format-name-n>;

Details The FORMAT statement changes a SAS variable format from its default format; the default SAS variable format is based on the data type and format of the PC file column. (See your PC file format chapter for information on the default data types and formats that the SAS System assigns to PC file data.)

An editing statement, such as FORMAT, must follow the CREATE statement and the database-description statements when you create a descriptor. See "CREATE" on page 17 for more information on the order of statements.

The *column-identifier* argument can be either the column name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the access descriptor. For example, to associate the DATE9. format with the BIRTHDATE column and with the second column in the access descriptor, submit the following statement:

```
format 2=date9. birthdate=date9.;
```

The column identifier is specified on the left and the SAS format is specified on the right of the expression. The equal sign (=) is optional. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotes. You can enter formats for as many columns as you want in one FORMAT statement.

You can use the FORMAT statement with a view descriptor only if the ASSIGN statement that was used when creating the access descriptor was specified with the **NO** value.

Note: When you use the FORMAT statement with access descriptors, the FORMAT statement also reselects columns that were previously dropped with the DROP statement. △

LIST

Lists columns in the descriptor and gives information about them.

Optional statement

Applies to: access descriptor or view descriptor

Default: ALL

Syntax

LIST <ALL|VIEW|<'>*column-identifier*<'> >;

Details The LIST statement lists columns in the descriptor along with information about the columns. You can use the LIST statement when creating an access descriptor or a view descriptor. The LIST information is written to your SAS log.

If you use an editing statement, such as LIST, it must follow the CREATE statement and the database-description statements when you create a descriptor. You can specify LIST as many times as you want while creating a descriptor; specify LIST last in your PROC ACCESS code to see the entire descriptor. Or, if you are creating multiple descriptors, specify LIST before the next CREATE statement in order to list all the information about the descriptor you are creating.

The LIST statement can take one or more of the following arguments:

ALL
lists all the columns in the PC file, the positional equivalents, the SAS variable names, and the SAS variable formats that are available for the access descriptor. When you are creating an access descriptor, ***NON-DISPLAY*** appears next to the column description for any column that has been dropped. When you are creating a view descriptor, ***SELECTED*** appears next to the column description for columns that you have selected for the view.

VIEW
lists all the columns that are selected for the view descriptor, along with their positional equivalents, their SAS names and formats, and any subsetting clauses. Any columns that were dropped in the access descriptor are not displayed. The VIEW argument is valid only for a view descriptor.

column-identifier
lists the specified column name, its positional equivalent, its SAS variable name and format, and whether the column has been selected. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotes.

The *column-identifier* argument can be either the column name or the positional equivalent, which is the number that represents the column's place in the descriptor. For example, to list information about the fifth column in the descriptor, submit the following statement:

```
list 5;
```

You can use one or more of these previously described options in a LIST statement, in any order.

PATH=

Specifies the path and filename of the file to be accessed.

Required statement

Applies to: access descriptor

Syntax

PATH= *'path-and-filename<.PC-file-extension>'* |
<'>filename<'> | *fileref* ;

Details The PATH= statement indicates the path and name of the file you want to access. The length of the filename and its other conventions can vary with the operating system. See the host documentation for your operating environment for more information.

For compatibility, place the PATH= statement immediately after the CREATE statement and before any other database-description statements when creating access descriptors. See "CREATE" on page 17 for more information.

You can specify PATH=statement with one of the following arguments:

'path-and-filename<.PC-file-extension>'
specifies the fully qualified path and filename. You must enclose the entire path and filename in quotes, including the appropriate PC file extension, such as .DBF,

.DIF, .WK1, .WK3, WK4, or .XLS. If you omit the file extension, SAS/ACCESS software supplies it for you.

<'>filename<'>
specifies the name of a file. The file must be located in your current (default) directory. If no extension is specified, the SAS/ACCESS interface supplies it for you. If the filename includes characters that are invalid in SAS names, such as the dollar sign ($) or if the filename begins with a number, you must enclose the entire filename in quotes.

fileref
specifies a fileref that references the path and name of the file. (Assigning filerefs with the FILENAME statement is described in *SAS Language and Procedures: Usage*.)

QUIT

Terminates the procedure.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

QUIT;

Details The QUIT statement terminates the ACCESS procedure without any further descriptor creation.

EXIT is the alias for the QUIT statement.

RENAME

Modifies the SAS variable name.

Optional statement

Applies to: access descriptor or view descriptor

Interacts with: ASSIGN, RESET

Syntax

RENAME *<'>column-identifier-1<'><=>*
SAS-variable-name-1
<...<'>column-identifier-n<'><=>
SAS-variable-name-n>;

Details The RENAME statement enters or modifies the SAS variable name that is associated with a column in a PC file. Use the RENAME statement when creating an access descriptor or a view descriptor.

An editing statement, such as RENAME, must follow the CREATE statement and the database-description statements when you create a descriptor. See "CREATE" on page 17 for more information on the order of statements.

Two factors affect the use of the RENAME statement: whether you specify the ASSIGN statement when you are creating an access descriptor, and the kind of descriptor you are creating.

- If you omit the ASSIGN statement or specify it with a NO value, the renamed SAS variable names that you specify in the access descriptor are retained throughout an ACCESS procedure execution. For example, if you rename the CUSTOMER column to CUSTNUM when you create an access descriptor, that column continues to be named CUSTNUM when you select it in a view descriptor unless a RESET statement or another RENAME statement is specified.

  When creating a view descriptor that is based on this access descriptor, you can specify the RESET statement or another RENAME statement to rename the variable again, but the new name applies only in that view. When you create other view descriptors, the SAS variable names are derived from the access descriptor variable names.

- If you specify the YES value in the ASSIGN statement, you can use the RENAME statement to change SAS variable names only while creating an access descriptor. As described earlier in the ASSIGN statement, SAS variable names and formats that are saved in an access descriptor are always used when creating view descriptors that are based on it.

The *column-identifier* argument can be either the PC file column name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the descriptor. For example, to rename the SAS variable names that are associated with the seventh column and the nine-character FIRSTNAME column in a descriptor, submit the following statement:

```
rename 7 birthdy  'firstname'=fname;
```

The column name (or positional equivalent) is specified on the left side of the expression, with the SAS variable name on the right side. The equal sign (=) is optional. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotes. You can rename as many columns as you want in one RENAME statement.

When you are creating a view descriptor, the RENAME statement automatically selects the renamed column for the view. That is, if you rename the SAS variable associated with a column, you do not have to issue a SELECT statement for that column.

When you are creating an access descriptor, the RENAME statement also reselects columns that were previously dropped with the DROP statement.

RESET

Resets PC file columns to their default settings.

Optional statement

Applies to: access descriptor or view descriptor

Interacts with: ASSIGN, DROP, FORMAT, RENAME, SELECT
Not allowed with: UPDATE

Syntax

RESET ALL | <'>*column-identifier-1*<'>
<...<'>*column-identifier-n*<'>>;

Details The RESET statement resets either the attributes of all the columns or the attributes of the specified columns to their default values. The RESET statement can be used when you create an access descriptor or a view descriptor, but it is not allowed when you are updating a descriptor. RESET has different effects on access and view descriptors, as described below.

If you use an editing statement, such as RESET, it must follow the CREATE statement and the database-description statements when you create a descriptor. See "CREATE" on page 17 for more information on the order of statements.

Access descriptors When you create an access descriptor, the default setting for a SAS variable name is a blank. However, if you have previously entered or modified any of the SAS variable names, the RESET statement resets the modified names to the default names that are generated by the ACCESS procedure. How the default SAS variable names are set depends on whether you included the ASSIGN statement. If you omitted ASSIGN or set it to NO, the default names are blank. If you set ASSIGN=YES, the default names are the first eight characters of each PC file column name.

The current SAS variable format is also reset to the default SAS format, which was determined from the column's data type. Any columns that were previously dropped, but that are specified in the RESET statement, become available; they can be selected in view descriptors that are based on this access descriptor.

View descriptors When you create a view descriptor, the RESET statement clears any columns that were included in the SELECT statement (that is, it de-selects the columns).

When creating the view descriptor, if you reset a SAS variable and then select it again within the same procedure execution, the SAS variable names and formats are reset to their default values, which are generated from the column names and data types. This applies only if you have omitted the ASSIGN statement or set the value to NO when you created the access descriptor on which the view descriptor is based. If you specified ASSIGN=YES when you created the access descriptor, the RESET statement has no effect on the view descriptor.

The RESET statement can take one or more of the following arguments:

ALL
for access descriptors, resets all the PC file columns that have been defined to their default names and format settings and reselects any dropped columns.

For view descriptors, ALL resets all the columns that have been selected so that no columns are selected for the view; you can then use the SELECT statement to select new columns. See "SELECT" on page 27 for more information on that statement.

column-identifier
can be either the PC file column name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the access descriptor. For example, to reset the SAS variable name and format associated with the third column, submit the following statement:

```
reset 3;
```

If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotes. You can reset as many columns as you want in one RESET statement, or use the ALL option to reset all the columns.

When creating an access descriptor, the *column-identifier* is reset to its default name and format settings. When creating a view descriptor, the specified column is no longer selected for the view.

SELECT

Selects PC file columns for the view descriptor.

Required statement

Applies to: view descriptor

Interacts with: RESET

Not allowed with: UPDATE

Syntax

SELECT ALL|<'>*column-identifier-1*<'>
<...<'>*column-identifier-n*<'>>;

Details The SELECT statement specifies which PC file columns in the access descriptor to include in the view descriptor. This is a required statement and is used only when you create view descriptors. You cannot use the SELECT statement when you are updating a view descriptor.

If you use an editing statement, such as SELECT, it must follow the CREATE statement when you create a view descriptor. See "CREATE" on page 17 for more information on the order of statements.

The SELECT statement can take one or more of the following arguments:

ALL
includes in the view descriptor all the columns that were defined in the access descriptor and that were not dropped.

column-identifier
can be either the column name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the access descriptor on which the view is based. For example, to select the first three columns, submit the following statement:

```
select 1 2 3;
```

If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotes. You can select as many columns as you want in one SELECT statement.

SELECT statements are cumulative within the same view creation. That is, if you submit the following two SELECT statements, columns 1, 5, and 6 are selected, not just columns 5 and 6:

```
select 1;
select 5 6;
```

To clear all your current selections when creating a view descriptor, use the RESET ALL statement; you can then use another SELECT statement to select new columns.

SUBSET

Adds or modifies selection criteria for a view descriptor.

Optional statement

Applies to: view descriptor

Syntax

SUBSET *selection-criteria*;

Details You use the SUBSET statement to specify selection criteria when you create a view descriptor. This statement is optional; if you omit it, the view retrieves all the data (that is, all the rows) in the PC file.

An editing statement, such as SUBSET, must follow the CREATE statement when you create a view descriptor. See “CREATE” on page 17 for more information on the order of statements.

UNIQUE

Generates SAS variable names based on PC file column names.

Optional statement

Applies to: view descriptor

Interacts with: ASSIGN

Not allowed with: UPDATE

Syntax

UNIQUE <=> YES|NO|Y|N ;

Details The UNIQUE statement specifies whether the SAS/ACCESS interface generates unique SAS variable names for PC file columns for which SAS variable names have not been entered. You cannot use the UNIQUE statement when you are updating a view descriptor.

An editing statement, such as UNIQUE, must follow the CREATE statement when you create a view descriptor. See “CREATE” on page 17 for more information on the

order of statements. The UNIQUE statement is affected by whether you specified the ASSIGN statement when you created the access descriptor on which this view is based, as follows:

- If you specified the ASSIGN=YES statement, you cannot specify UNIQUE when creating a view descriptor. YES causes the SAS System to generate unique names, so UNIQUE is not necessary.
- If you omitted the ASSIGN statement or specified ASSIGN=NO, you must resolve any duplicate SAS variable names in the view descriptor. You can use UNIQUE to generate unique names automatically, or you can use the RENAME statement to resolve duplicate names yourself. See "RENAME" on page 24 for information on that statement.

If duplicate SAS variable names exist in the access descriptor on which you are creating a view descriptor, you can specify UNIQUE to resolve the duplication. When you specify UNIQUE=YES, the SAS/ACCESS interface appends numbers to any duplicate SAS variable names, thus making each variable name unique. (See "CREATE" on page 17 for an explanation of how to create descriptors.)

If you specify UNIQUE=NO, the SAS/ACCESS interface continues to allow duplicate SAS variable names to exist. You must resolve these duplicate names before saving (and thereby creating) the view descriptor.

Note: It is recommended that you use the UNIQUE statement. If you omit it and the SAS System encounters duplicate SAS variable names in a view descriptor, your job fails.

The equals (=) sign is optional in the UNIQUE statement. UN is the alias for UNIQUE. △

UPDATE

Updates a SAS/ACCESS descriptor file.

Optional statement

Applies to: access descriptor or view descriptor

Not allowed with: ASSIGN, RESET, SELECT, UNIQUE

Syntax

UPDATE *libref.descriptor-name*.ACCESS | VIEW ;

Details Use the UPDATE statement to perform a quick, simple update of a descriptor. For example, if the PC database file for an existing access descriptor is relocated, you can use UPDATE with the PATH option to specify the new location.

Descriptors modified by UPDATE are not checked for errors. Where validation is crucial, use CREATE to overwrite a descriptor rather than UPDATE.

The descriptor is a name in three parts separated by periods (.):

libref
identifies the library container, which is a location either on the local system's disk or that the local system can directly access. The *libref* must have been previously created by a LIBNAME statement.

descriptor-name
is the descriptor you are updating. It must already exist in *libref*. (See CREATE for an explanation of how to create descriptors.)

ACCESS
indicates you are updating an access descriptor while VIEW indicates you are updating a view descriptor.

Multiple UPDATE statements may appear in one ACCESS procedure block. If you use UPDATE to change an access descriptor, one or more UPDATE statements may be required for views that depend on the modified access descriptor.

You can use UPDATE and CREATE in the same PROC ACCESS block.

Updating access descriptors

The order of statements in an UPDATE block is as follows:

1. UPDATE must be the first statement after the PROC ACCESS statement with one exception: if the block includes both UPDATE and CREATE statements, either statement may be the first in the block.
2. DBMS description statements are next. All are allowed.
3. Editing statements are next. These editing statements are not allowed: ASSIGN, LIST, RESET, SELECT, VIEW.

Since the UPDATE block does not validate the updated descriptor, the order of description and editing statements does not matter.

Updating view descriptors

1. UPDATE must be the first statement after the PROC ACCESS statement with one exception: if the block includes both UPDATE and CREATE statements, either statement may be the first in the block.
2. DBMS description statements are next. All are allowed.
3. Editing statements are next. These editing statements are not allowed: ASSIGN, DROP, RESET, SELECT, and UNIQUE.

Examples

The following example upates an existing access descriptor named ADLIB.PRODUCT:

```
libname adlib 'c:\sasdata';

proc access dbms=wk4;
   update adlib.product.access;
   path=c:\lotus\specprod.wk4;
   rename productid prodid
          fibername fiber;
   format productid  4.
          weight     e16.9
          fibersize  e20.13
          width      e16.9;
run;
```

The following example updates the access descriptor EMPLOY, located in **c:\sasdata**, for the spreadsheet named **c:\excel\employ.xls** and updates the view descriptor for EMPLOY named EMP1204, located in **c:\sasviews**:

```
libname adlib 'c:\sasdata';
libname vlib 'c:\sasviews';

proc access dbms=xls;
   update adlib.employ.access;
   path='c:\excel\employ.xls';
   list all;

   update vlib.emp1204.view;
   format empid 6.
          salary dollar12.2
          jobcode 5.
          hiredate datetime9.
          birthdate datetime9.;
   subset where jobcode=1204;
run;
```

The following example updates a second view descriptor that is based on EMPLOY named BDAYS. It is also located in **c:\sasviews**. When you update a view, it is not necessary to specify the access descriptor (using ACCDESC=) in the PROC ACCESS statement. Note that FORMAT can be used because the access descriptor EMPLOY was created with ASSIGN=NO.

```
libname vlib 'c:\sasviews';

proc access dbms=xls;
   update vlib.bdays.view;
   format empid 6.
          birthdate datetime7.;
run;
```

Performance and Efficient View Descriptors

General Guidelines

When you create and use view descriptors, follow these guidelines to minimize the use of SAS System resources and to reduce the time it takes to access data:

- Select only the columns your SAS program needs. Selecting unnecessary columns adds extra processing time.
- Where possible, specify selection criteria to subset the number of observations processed by the SAS System.
- To present PC data in sorted order, reference a view descriptor in a PROC SQL query. Otherwise, you may need to extract the data to sort it, as described below.

Extracting Data Using a View

In some cases, it might be more efficient to use a view descriptor to extract PC file data and place it in a SAS data file instead of using the view descriptor to read the data directly.

A PC file is read every time a view descriptor is referred to in a SAS program and the program is executed; the program's output reflects the latest updated level of the

PC file. If many users are reading the same PC file repeatedly, PC file performance may decrease. If you create several reports during the same SAS session, they may not be based on the same PC file data due to updating by other users. Therefore, in the following circumstances, it is better to extract data:

- Extract PC file data if the file is large and you use the data repeatedly in SAS programs.

  If a view descriptor describes a large PC file and you plan to use the same PC file data in several procedures or DATA steps during the same SAS session, you might improve performance by extracting the data. Placing the data into a SAS data file requires a certain amount of disk space to store the data and I/O to write the data. However, SAS data files are organized to provide optimal performance with PROC and DATA steps. Programs that use SAS data files are often more efficient than programs that read PC file data directly.

- Extract PC file data if you use sorted data several times in a SAS program.

  If you intend to use PC file data in a particular sorted order several times, run the SORT procedure on the view descriptor using the OUT= option to extract the data. The OUT= option is required whenever PROC SORT references a view descriptor. Extracting the data in this way is more efficient than requesting the same sort repeatedly on the PC file data. Note that using the ORDER BY clause in the SQL procedure does not sort the data in the physical PC file; it only presents the data in a sorted order.

- Extract PC file data for added security.

  If you are the owner of a PC file and do not want anyone else to read the data, you might want to extract the data (or a subset of the data) and not distribute information about either the access descriptor or view descriptor. Or, you might want to assign PC file security features to your PC files to prevent unauthorized reading or writing to them.

  On the SAS System side, you might also want to assign SAS System passwords to your descriptors for additional security. If a view descriptor has a password assigned to it and you extract the data, the new SAS data file is automatically assigned the same password. If a view descriptor does not have a password, you can assign a password to the extracted SAS data file.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Software for PC File Formats: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Software for PC File Formats: Reference, Verison 8

ISBN 1-58025–544–2

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

1st printing, October 1999