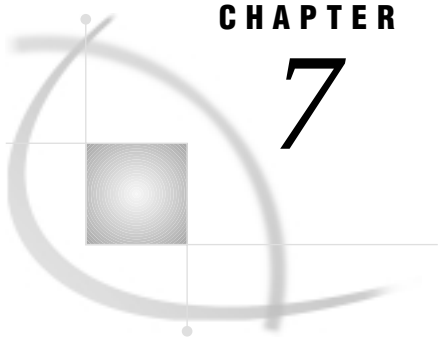

CHAPTER

7

ACCESS Procedure Reference

Introduction

The ACCESS procedure enables you to create and edit the descriptor files used by the SAS/ACCESS interface to SYSTEM 2000 software.

This chapter provides reference information for the ACCESS procedure and its options and statements. First, the PROC ACCESS statement is presented, then each of the options and statements. For examples of how to use the statement options, refer to Chapter 3, “Defining SAS/ACCESS Descriptor Files,” on page 19.

“Creating and Using View Descriptors Efficiently” on page 95 presents several efficiency considerations for using the SAS/ACCESS interface to SYSTEM 2000 software. The final section, “ACCESS Procedure Data Conversions” on page 96, summarizes how the SAS/ACCESS interface treats each type of SYSTEM 2000 data.

For information on SAS data sets and data libraries and their naming conventions, or if you need help with the terminology used in this procedure description, refer to the *SAS Language Reference: Dictionary* and the SAS documentation for your host system. Help is also available from within the SAS System by choosing Help then

SAS System Help from the menu bar, then Help on SAS System Products and SAS/ACCESS in the help system.

Syntax

PROC ACCESS <*options*>;

CREATE *libref.member-name*.ACCESS | VIEW;
DATABASE<=>*database-name* <**S2KPW**<=>*password*
<**MODE**<=>SINGLE | MULTI>>;
ASSIGN<=>YES | NO;
BYKEY *variable-identifier*<=>YES | NO
<...*variable-identifier-n*<=>YES | NO>;
DROP *variable-identifier*
<...*variable-identifier-n*>;
FORMAT *variable-identifier*<=>*SAS-format-name*
<...*variable-identifier-n*<=>*SAS-format-name-n*>;
INFORMAT *variable-identifier*<=>*SAS-informat-name*
<...*variable-identifier-n*<=>*SAS-informat-name-n*>;
LENGTH *variable-identifier*<=>*item-width*
<...*variable-identifier-n*
<=>*item-width-n*>;
LIST <ALL | VIEW | *variable-identifier*>;
QUIT;
RENAME *variable-identifier*<=>*SAS-variable-name*
<...*variable-identifier-n*<=>*SAS-variable-name-n*>;
RESET ALL | *variable-identifier* <...*variable-identifier-n*>;
SELECT ALL | *variable-identifier* <...*variable-identifier-n*>;
SUBSET *selection-criteria*;
S2KPW<=>*password* <**MODE** <=>MULTI | SINGLE>;
UNIQUE<=>YES | NO;

PROC ACCESS Statement Options

The following options can be used with the PROC ACCESS statement.

ACCDESC= libref.access-descriptor
identifies an access descriptor. You use this option to create a view descriptor from an existing access descriptor.

If the access descriptor has been assigned a SAS password, you may need to specify the password in the ACCDESC= option in order to create a view descriptor based on the access descriptor. Whether you specify the password depends on the level of protection that was assigned to the access descriptor. For information on assigning and using passwords, see “Specifying Passwords for SAS/ACCESS Descriptors” on page 75.

If you create the access descriptor and the view descriptor in the same execution of PROC ACCESS, omit the ACCDESC= option because you specify the access descriptor’s name in the CREATE statement.

AD= and ACCESS= are aliases for this option.

DBMS= S2K
specifies that you want to invoke the SAS/ACCESS interface to SYSTEM 2000 software. This option is required when creating a descriptor, but not when extracting DBMS data.

OUT=libref.member
specifies the SAS data file to which DBMS data are written. The OUT= option is used only with the VIEWDESC= option.

VIEWDESC=*libref.view-descriptor*
specifies a view descriptor that accesses the DBMS data. VIEWDESC= is used only with the OUT= option.
VIEW= and VD= are aliases for this option.

See Appendix 3, "Example Data," on page 143 for examples of using options in procedure statements.

Specifying Passwords for SAS/ACCESS Descriptors

The SAS/ACCESS interface requires both access descriptors and view descriptors to have a SYSTEM 2000 password to access the database. The password for the access descriptor determines the picture of the database used to create view descriptors. The password for the view descriptor determines the data you see, and your ability to subset and edit it through the descriptor.

You specify the password for the access descriptor as part of the DATABASE statement. For the view descriptor, you have the option to store the SYSTEM 2000 password in the view descriptor (S2KPW statement) or submit it as a SAS data set option. Storing the SYSTEM 2000 password in a view descriptor enables all who use it to have access to its data. Relying on the data set option means giving users access to database passwords.

To protect your database passwords, you may want to store the SYSTEM 2000 password in the view descriptor and assign one or more SAS passwords to control access to the descriptor file. You can also assign SAS passwords to control who can create view descriptors from an access descriptor. To access the descriptor files, you specify the SAS password as a data set option. For example, to create a view descriptor, you would specify the access descriptor password after the ACCDESC= option as follows:

```
proc access dbms=s2k accdesc=mylib.employee
  (alter=reward);
   create vlib.customer.view;
   select all;
run;
```

You must first create descriptor files before assigning SAS passwords to them.

Table 7.1 on page 76 summarizes the levels of protection that SAS System passwords have and their effects on descriptor files.

Table 7.1 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 DBMS data from being read or updated	protects DBMS data from being updated	protects descriptor from being read or edited

For detailed information about the levels of protection and the types of SAS passwords you can use, refer to *SAS Language Reference: Dictionary*. For more information about protecting access and view descriptors, see "Ensuring Data Security" on page 123. The following section describes how you assign SAS System passwords to descriptors.

Assigning SAS System Passwords

You can assign, change, or clear a password for an access descriptor, a view descriptor, or another SAS file in the SAS System by using 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 REWARD with the ALTER level of protection to the access descriptor MYLIB.EMPLOYE:

```
proc datasets library=mylib memtype=access;
   modify employe (alter=reward);
run;
```

In this case, users are prompted for the password whenever they try to browse or edit the access descriptor or to create view descriptors that are based on MYLIB.EMPLOYE.

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.CUSTACCT:

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

In this case, users are prompted for the SAS password when they try to read the DBMS data, or try to browse or edit the view descriptor VLIB.CUSTACCT itself. You need both levels to protect the data and descriptor from being read. However, a user could still update the data accessed by VLIB.CUSTACCT, for example, by using a PROC SQL UPDATE. 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 custacct (read=mypw/ alter=mydept/);
run;
```

In the following example, PROC DATASETS sets a READ and ALTER password for view descriptor VLIB.CUSTINFO. PROC PRINT tries to use the view descriptor with both an invalid and valid password.

```
/* Assign passwords */
proc datasets library=vlib memtype=view;
  modify custinfo (read=r2d2 alter=c3po);
run;

/* Invalid password given */
proc print data=vlib.custinfo (pw=r2dq);
  where ssn = '178-42-6534';
  title2 'Data for 178-42-6534';
run;

/* Valid password given */
proc print data=vlib.custinfo (pw=r2d2);
  where ssn = '178-42-6534';
  title2 'Data for 178-42-6534';
run;
```

Refer to *SAS Language Reference: Dictionary* for more examples of assigning, changing, deleting, and using SAS System passwords.

Procedure Statements

Within SAS/ACCESS software, there are two categories of procedure statements: database-description statements and editing statements. In SAS/ACCESS interface to SYSTEM 2000 software, the DATABASE statement and its options describe the database. All other statements, except CREATE, are considered to be editing statements and are optional. The DATABASE statement is specified after the CREATE statement but before any editing statements.

The options and statements you use with the ACCESS procedure depends on your task.

For example, to create an access descriptor:

```
proc access dbms=s2k;
   create mylib.employe.access;
      DATABASE statement;
      optional editing statement(s);
run;
```

To create an access descriptor and a view descriptor:

```
proc access dbms=s2k;
   create mylib.employe.access;
      DATABASE statement;
      optional editing statement(s);

   create vlib.emppos.view;
      optional editing statement(s);
run;
```

To create a view descriptor from an existing access descriptor:

```
proc access dbms=s2k accdesc=mylib.employe;
   create vlib.emppos.view;
      optional editing statement(s);
run;
```

The procedure statements are described in the following sections.

ASSIGN=

Generates SAS names and formats that are based on item names and data types

Optional statement

Applies to: access descriptor

Syntax

ASSIGN=|AN= YES|NO|Y|N;

Details The ASSIGN= statement causes view descriptors to inherit the SAS variable names and formats of the parent access descriptor at the time that the access descriptor is created. That is, if ASSIGN=Y, the variable names generated for the access descriptor will be used in all derived view descriptors, regardless of the statements used in the view descriptor. If ASSIGN=NO (or N), which is the default value, you specify the SAS variable names and formats when you create a view descriptor from this access descriptor. You use the RENAME, FORMAT, INFORMAT, LENGTH, BYKEY, and UNIQUE statements to change the variable names and attributes during a descriptor's creation.

The ASSIGN= statement generates SAS variable names based on the first eight, non-blank characters of the item names and SAS variable attributes based on the items' data types. You can change the names and formats, but only in the access descriptor. The names saved in the access descriptor are the ones that will be used in the view descriptors.

When a new CREATE statement is entered, the ASSIGN= statement is reset to the default NO value.

AN is the alias for the ASSIGN statement.

BYKEY

Designates one or more items as sort keys

Optional statement

Applies to: access descriptor and view descriptor

Syntax

BYKEY *variable-identifier*<=> YES|NO

<...*variable-identifier-n*<=> YES|NO>;

Details The BYKEY statement designates one or more items as BY keys and in a view descriptor, also selects them for the view.

The BYKEY statement cannot be used to change the BYKEY value in a view descriptor if the ASSIGN= statement in the access descriptor from which the view descriptor is derived has a value of YES (or Y).

The BYKEY statement applies to data items. The *variable-identifier* argument can be one of the following:

- current SAS name for the data item

  Note: Any name on the lefthand side of the equal sign must be a SAS name, not a SYSTEM 2000 name. In an access descriptor, if the ASSIGN statement is omitted, you must use the item number or component number (C-number) on the lefthand side of the equal sign. △
- positional equivalent, which is the number that represents the item, as given by the LIST statement
- SYSTEM 2000 C-number of the database item.

For example, if you want to make the third item a BY key, issue the following statement:

```
bykey 3=y;
```

CREATE

Creates an access descriptor or view descriptor

Required statement

Applies to: access descriptor and view descriptor

Syntax

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

Details The CREATE statement identifies an access descriptor or view descriptor that you want to create.

To create the descriptor, use a three-level name. The first level is the libref of the SAS data library where you want the descriptor stored. You can store the descriptor in a temporary (WORK) or permanent SAS data library. The second level is the access descriptor's name (that is, the member name). The third level is the type of SAS file: ACCESS for access descriptors and VIEW for view descriptors.

You can create access descriptors and view descriptors in the same procedure statement (view descriptors directly following the access descriptors that they describe), unless you specify the ACCDESC= option in the PROC ACCESS statement. Then, the CREATE statement will create only view descriptors.

When you submit a CREATE statement for processing, the SAS/ACCESS interface checks the statement for errors. The descriptor is not actually written until the next CREATE or RUN statement is processed. If the SAS/ACCESS interface finds errors,

error messages are written to the SAS log and processing is terminated. After you correct the error, resubmit the statements for processing.

The database-identification and DROP statements cannot be specified when creating a view descriptor.

DATABASE

Specifies the SYSTEM 2000 database to use

Required statement

Applies to: access descriptor

Syntax

DATABASE<=>*database-name* **S2KPW**<=>*password* <**MODE**<=>*access-mode*>;

Details The DATABASE statement specifies the name of the SYSTEM 2000 database that you want to access, its password, and optionally the access mode. The DATABASE statement should immediately follow the CREATE statement for the access descriptor being created.

The database name can be one to 16 characters long. Names longer than 16 characters will be truncated without an error message. If the database name contains blanks or special characters, enclose the name in single or double quotes.

The database password, submitted with the S2KPW argument, determines the picture of the database that can be used to create view descriptors. For information about acceptable passwords, see “SYSTEM 2000 Passwords” on page 17.

The syntax for the S2KPW argument is:

S2KPW<=>*password*

The password can be one to four characters long, with no embedded blanks, and optionally enclosed in single quotes. Passwords longer than four characters will be truncated with a warning message. If you specify a special character for a password, it must be a single character (that is, a one-character password) and enclosed in single quotes. This argument is required.

The syntax for the MODE argument is:

<**MODE**<=> SINGLE|MULTI>

The MODE argument specifies your mode of accessing SYSTEM 2000 software. SINGLE means the database is in a single-user environment (that is, a database in your SAS program environment). MULTI means the database files are in the Multi-User environment. The default value is MULTI. The SINGLE value can be abbreviated as SU or S. The MULTI value can be abbreviated as MU or M.

DB, DBN, and S2KDB are aliases for the DATABASE statement.

DROP

Drops the specified item so that it is not available for selection

Optional statement

Applies to: access descriptor

Syntax

DROP *variable-identifier* <*...variable-identifier-n*>;

Details The DROP statement drops the specified variable from the access descriptor so that the variable is not available for selection when creating a view descriptor. The specified variable in the database remains unaffected by this statement.

If you drop a record, every item in the record is dropped.

The *variable-identifier* argument can be one of the following:

- current SAS name for the item
- positional equivalent, which is the number that represents the item, as given by the LIST statement
- SYSTEM 2000 C-number of the database item.

For example, if you want to drop the third and fifth items, submit the following statement:

```
drop 3 5;
```

If you are creating an access descriptor in interactive line mode and want to mark an item as display that was previously marked as non-display with the DROP statement, use the RESET statement for that item. Note, however, that this will also reset the various attributes of that item to their default values (such as name, format, and so on).

FORMAT

Assigns a SAS format to a SYSTEM 2000 data item

Optional statement

Applies to: access descriptor and view descriptor

Syntax

FORMAT *variable-identifier* <=> *SAS-format-name*
<*...variable-identifier-n*<=>*SAS-format-name-n*>;

Details The FORMAT statement changes a SAS variable format from its default format; the default format is based on the database item's data type. You can enter formats for as many items as necessary using one FORMAT statement.

You cannot specify the FORMAT statement for a record.

The *variable-identifier* argument can be one of the following:

- current SAS variable name for the item

  Note: Any name on the lefthand side of the equal sign must be a SAS name, not a SYSTEM 2000 name. In an access descriptor, if the ASSIGN statement is

omitted, you must use the item number or component number (C-number) on the lefthand side of the equal sign. △

- positional equivalent, which is the number that represents the item, as given by the LIST statement
- SYSTEM 2000 C-number of the database item.

For example, if you want to associate the DATE9. format with the fifth item in the access descriptor, issue the following statement:

```
format 5 date9.;
```

You can only use the FORMAT statement with a view descriptor if the ASSIGN statement used when creating the access descriptor was specified with the NO value. When used in a view descriptor, the FORMAT statement automatically selects the reformatted item. That is, if you change the format associated with an item, you do not have to issue a SELECT statement for that item.

FMT is an alias for the FORMAT statement.

INFORMAT

Assigns a SAS informat to a SYSTEM 2000 item

Optional statement

Applies to: access descriptor and view descriptor

Syntax

INFORMAT *variable-identifier<=> SAS-informat-name <...variable-identifier-n<=> SAS-informat-name-n>*;

Details The INFORMAT statement changes a SAS variable informat from its default informat; the default informat is based on the database item's data type. You can enter as many informats as necessary using one INFORMAT statement.

You cannot specify the INFORMAT statement for a record.

The *variable-identifier* argument can be one of the following:

- current SAS variable name for the item

  Note: Any name on the lefthand side of the equal sign must be a SAS name, not a SYSTEM 2000 name. In an access descriptor, if the ASSIGN statement is omitted, you must use the item number or component number (C-number) on the lefthand side of the equal sign. △
- positional equivalent, which is the number that represents the item, as given by the LIST statement
- SYSTEM 2000 C-number of the database item.

For example, if you want to associate the DATE7. informat with the second item in the access descriptor, issue the following statement:

```
informat 2 DATE7.;
```

You can only use the INFORMAT statement with a view descriptor if the ASSIGN statement in the access descriptor from which it is derived is specified with the NO

value. When used for a view descriptor, the INFORMAT statement automatically selects the reformatted item. That is, if you change the informat associated with an item, you do not have to issue a SELECT statement for that item.

INF is an alias for the INFORMAT statement.

LENGTH

Assigns a character width to a data item

Optional statement

Applies to: access descriptor and view descriptor

Syntax

LENGTH *variable-identifier*<=> *item-width*
<...*variable-identifier-n*<=> *item-width-n*>;

Details The LENGTH statement changes the item width in characters from the default width; the default item width is based on the database item's picture specification. This statement enables the SAS System to deal with S2K CHARACTER/TEXT items that overflow their widths (the SAS System does not permit variable-length character variables). The *item-width* argument can be no greater than 200.

The LENGTH statement only applies to data items; you cannot specify a length for a record.

The *variable-identifier* argument can be one of the following:

- current SAS name for the item

  Note: Any name on the lefthand side of the equal sign must be a SAS name, not a SYSTEM 2000 name. In an access descriptor, if the ASSIGN statement is omitted, you must use the item number or component number (C-number) on the lefthand side of the equal sign. △
- positional equivalent, which is the number that represents the item, as given by the LIST statement
- SYSTEM 2000 C-number of the database item.

You can only use the LENGTH statement with a view descriptor if the ASSIGN statement in the access descriptor from which it is derived is specified with the NO value. When used for a view descriptor, the LENGTH statement automatically selects the reformatted item. That is, if you change the length associated with an item, you do not have to issue a SELECT statement for that item.

LEN and S2KLEN are aliases for the LENGTH statement.

LIST

Lists all or selected items in the descriptor and information about the items

Optional statement

Applies to: access descriptor and view descriptor

Syntax

LIST <ALL | VIEW | *variable-identifier*>;

Details The LIST statement lists all or selected items in the descriptor and attributes of items, including their positional equivalents, SYSTEM 2000 component numbers, default SAS variable names based on the first eight, non-blank characters of the SYSTEM 2000 item names, and the default SAS formats based on the SYSTEM 2000 data types.

Note: The SYSTEM 2000 item names are not listed in the log because of their possible 40 or more character length. △

The LIST information is written to your SAS log.
The LIST statement can take one or more of the following arguments:

ALL — lists all items and item attributes available in the access descriptor for selection. If an item has been dropped when creating an access descriptor, *NON-DISPLAY* is shown next to the item's description. When creating a view descriptor, items selected for the view are shown with *SELECTED* next to the item's description.

If you do not specify an argument, the default is ALL.

VIEW — lists all items and item attributes in the access descriptor selected for the view descriptor and any subsetting or ordering criteria. The VIEW argument is only valid when creating a view descriptor.

variable-identifier — the current SAS name, the positional equivalent (which is the number that represents the item, as given by the LIST statement), or the SYSTEM 2000 C-number of the database item.

If you specify a record in a LIST statement, all the data items in that record are listed.

For example, if you want to list information about the fifth item in the database, issue the following statement:

```
list 5;
```

If you want to list all of the items in the database followed by the items selected for the view descriptor, issue the following statement:

```
list all view;
```

QUIT

Terminates the procedure without any further descriptor creation

Optional statement

Applies to: access descriptor and view descriptor

Syntax

QUIT|EXIT;

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

EXIT is the alias for the QUIT statement.

RENAME

Enters or modifies the SAS name for an item

Optional statement

Applies to: access descriptor and view descriptor

Syntax

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

Details The RENAME statement enters or modifies the SAS variable name associated with a database item. If you are creating a view descriptor from an existing access descriptor that has an ASSIGN value of YES (or Y), you cannot use the RENAME= statement.

When creating an access descriptor and ASSIGN=YES, you can use the RENAME statement to assign new SAS names to the default SAS names and these new names will always be used when creating view descriptors based on the access descriptor.

When ASSIGN=NO, any names assigned in the access descriptor can be changed in the view descriptor with the RENAME statement, but the new name applies only in that view.

The *variable-identifier* argument can be one of the following:

- current SAS variable name for the item

  Note: Any name on the lefthand side of the equal sign must be a SAS name, not a SYSTEM 2000 name. In an access descriptor, if the ASSIGN statement is omitted, you must use an item number or component number (C-number) on the lefthand side of the equal sign. △

- positional equivalent from the LIST statement, which is the number that represents the item's place in the descriptor
- SYSTEM 2000 C-number of the database item.

For example, if you want to modify the SAS variable names associated with the fourth and fifth items in a descriptor, issue the following statement:

```
rename 4=hire birthday=birth;
```

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

RESET

Resets specified or all items to their default settings

Optional statement

Applies to: access descriptor and view descriptor

Syntax

RESET ALL | *variable-identifier*
<*...variable-identifier-n*>;

Details The RESET statement resets all or the specified items to their default values.

When creating an access descriptor, the default setting for a SAS variable name is a blank, unless you enter SAS variable names using the RENAME statement or include the ASSIGN=YES statement. Therefore, when using the RESET statement, the SAS variable names can be reset to the default name generated by the ACCESS procedure (that is, the first eight characters of the variable name) or to a blank. Items dropped with a DROP statement also become available and can be selected in view descriptors based on this access descriptor.

When creating a view descriptor, the results depend on the setting of the ASSIGN statement in the access descriptor on which the view descriptor is based. If ASSIGN=Y, the RESET statement cannot be used in the view descriptor. If ASSIGN=NO, if you reset SAS variable names and variable attributes and then select them later within the same procedure execution, the SAS variable names and attributes are reset to the default values generated from the item names and data types. In a view descriptor, the RESET statement clears any items specified in the SELECT statement (that is, it unselects the items).

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

ALL — resets all the database items defined in the access descriptor to their default name and attribute settings. When creating a view descriptor, the ALL argument resets all the items that have been selected, so that no items are selected for the view; you can then use the SELECT statement to select new items. See the SELECT statement later in this chapter for more information.

variable-identifier — can be the current SAS name, the positional equivalent (which is the number that represents the item as given by the LIST statement), or the SYSTEM 2000 component number of the database item.

For example, if you want to reset the SAS variable name and attribute associated with the third item, issue the following statement:

```
reset 3;
```

SELECT

Selects the items in the access descriptor that are to be included in the view descriptor

Optional statement
Applies to: view descriptor

Syntax

SELECT ALL | *variable-identifier*
<...*variable-identifier-n*>;

Details The SELECT statement selects the database items in the access descriptor to be included in the view descriptor.

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

ALL
includes in the view descriptor all of the items defined in the access descriptor that were not dropped.

variable-identifier
can be the current SAS name, the positional equivalent (which is the number that represents the item as given by the LIST statement), or the SYSTEM 2000 component number of the database item.

For example, if you want to select the first three items, issue the following statement:

```
select 1 2 3;
```

You can select as many items as you want using one SELECT statement.

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

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

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

Selecting a record selects all items within the record.

SUBSET

Adds or modifies selection criteria defined for a view descriptor

Optional statement
Applies to: view descriptor

Syntax

SUBSET *selection-criteria*;

Details The SUBSET statement specifies the selection criteria when creating a view descriptor This statement is optional, but omitting it causes the view to retrieve all the data in the database. For example, you can issue the following statement:

```
subset "where amount<1010";
```

If you have multiple selection criteria, enter them all in one SUBSET statement, as in the following example:

```
subset "where amount<1010"
       "ob amount";
```

The quoted strings are concatenated and passed to SYSTEM 2000 software for processing.

For more information on selection and ordering criteria for the SAS/ACCESS interface to SYSTEM 2000 data management software, refer to your Release 6.06 SAS/ACCESS documentation.

To clear the selection criteria, issue a SUBSET statement without an argument, as follows:

```
subset;
```

S2KPW

Stores the SYSTEM 2000 password and access mode for a view descriptor

Optional statement

Applies to: view descriptor

Syntax

S2KPW<=>*password* <MODE<=>MULTI|SINGLE>;

Details The S2KPW statement specifies a SYSTEM 2000 password and optional access mode for creating a view descriptor. The password you specify will be stored in encrypted form and enable all who access the view descriptor to have access to the data it describes. If you do not specify the S2KPW statement when creating a view descriptor, you must specify the password when using the view descriptor to access data from the database.

The password used when you open a view descriptor determines the data you see and your ability to subset and edit it through the view descriptor. You can specify the password used in the access descriptor from which the view is derived, or another password that encompasses a subset of its data. If you specify a password that does not encompass data from the access descriptor, the view will be created, but the software will issue an error message when you attempt to open it.

The password specified in the S2KPW statement can be one to four characters long, with no embedded blanks, and optionally enclosed in single quotes. Passwords longer than four characters will be truncated with a warning message. If you specify a special character for a password, it must be a single character (that is, a one-character password) and enclosed in single quotes.

The S2KPW statement takes one optional argument, as follows:

MODE <=> SINGLE|MULTI
specifies your mode of accessing SYSTEM 2000 software. SINGLE means the database is in a single-user environment (that is, a database in your SAS program environment). MULTI means the database files are in the Multi-User environment. The default value is MULTI. The SINGLE value can be abbreviated as SU or S. The MULTI value can be abbreviated as MU or M.

MD, S2KMD, and S2KMODE are aliases for the MODE argument. The mode is also stored with the view.

UNIQUE

Generates unique SAS names based on item names

Optional statement

Applies to: view descriptor

Syntax

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

Details The UNIQUE statement specifies whether the SAS/ACCESS interface should generate unique SAS variable names for items for which SAS variable names or variable attributes have not been entered.

Use of the UNIQUE statement is affected by whether you specified the ASSIGN statement when creating the access descriptor on which this view is based.

- If you specified the ASSIGN statement with a YES value, you cannot specify the UNIQUE statement when creating a view. The YES value causes the SAS System to generate unique names, so the UNIQUE statement is not necessary.
- If you omitted the ASSIGN statement or specified it with a NO value, you must resolve any duplicate SAS variable names in the view descriptor. You can use the UNIQUE statement to automatically generate unique names, or you can use the RENAME statement to resolve these duplicate names yourself. See the RENAME statement earlier in this chapter for information on it.

If duplicate SAS variable names exist in the access descriptor on which you are creating a view descriptor, you can specify the UNIQUE statement to resolve the duplication. You specify the YES (or Y) value to have the SAS/ACCESS interface append numbers to any duplicate SAS variable names, thus making each variable name unique.

If you specify a NO (or N) value for the UNIQUE statement, 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.

If you are running your SAS/ACCESS job in noninteractive or batch mode, it is recommended that you use the UNIQUE statement. If you do not and the SAS System encounters duplicate SAS variable names in a view descriptor, your job will fail.

UN is the alias for this statement.

SYSTEM 2000 Where-Clause

Use a SYSTEM 2000 where-clause to select particular logical entries from a SYSTEM 2000 database. You may reference any item included in the access descriptor on which the view descriptor is based, as long as the password you are using has where-clause authority for each referenced item.

When you include a SYSTEM 2000 where-clause in a view descriptor, the selection criteria are executed each time you use the view descriptor in a SAS program. When a SYSTEM 2000 where-clause is invoked, the interface view engine

- replaces references to SAS variable names with database item component numbers. (The SAS variable names must correspond to a database item included in the view descriptor.)
- translates keywords to uppercase for compatibility with SYSTEM 2000 software.
- expands connecting strings to connect the SAS WHERE clause to the view where-clause.
- preserves significant blanks in delimited textual values.

The syntax of the where-clause can include one or more of the following conditions. Examples of these conditions are presented in “Examples” on page 92.

Note: This is a partial description of the SYSTEM 2000 where-clause. For a complete description, see the *SYSTEM 2000 QUEST Language* manual. However, you cannot include a Collect File item name or the SAME operator in a where-clause included in a view descriptor. △

Syntax

WHERE *expression*;

WHERE
is the keyword designating a where-clause. You can also use the abbreviation WH. The keyword is optional if the where-clause is the first clause or if you do not specify an ordering-clause.

expression
consists of one of the following:

| *condition*
|(*expression*)
|NOT *expression*
|*expression* AND *expression*
|*expression* OR *expression*
|*record* HAS *expression*
|*expression* AT *n*

condition [NON-KEY] *item*
|unaryoperator
|binaryoperator *value*
|ternaryoperator *value* * *value*
| CONTAINS *text*
|* binaryoperator *item**

NON-KEY
allows you to change a key condition to a non-key one. This capability is not available in a SAS WHERE clause. See "Using HAS, AT, and NON-KEY" on page 141 for information on using connecting strings to extend the function of the NON-KEY specification to the SAS WHERE clause conditions.

You can abbreviate NON-KEY to NK.

NOT
finds the complement of specified criteria. You can also use the ¬ symbol.

AND
combines two expressions by finding data records that satisfy both expressions. You can also use the & symbol.

OR
combines two expressions by finding data records that satisfy either expression or both. You can also use the | symbol.

record
is a schema record name or component number.

HAS
specifies a data record by its position under its parent. This capability is not available in a SAS WHERE clause. See "Using HAS, AT, and NON-KEY" on page 141 for information on using connecting strings to extend the function of the AT operator to the SAS WHERE clause conditions.

n
is 0 or a positive integer indicating position of a record under its parent. Zero means the last position.

item
is a schema item name or component number included in the access descriptor. Or you can specify a SAS variable name if the item is included in the view descriptor. The item can be key or non-key.

unary-operator: EXISTS or FAILS
specifies the existence or nonexistence of values. You can also specify EXIST or EXISTING and FAIL or FAILING.

binary-operator: EQ, NE, GE, GT, LE, or LT
compares an item with a value or compares two items. You can also use these symbols:

Table 7.2

Operator	Alternate Form
EQ	=
NE	¬= or !=
GE	>= or => or ¬< or !<
GT	>

Operator	Alternate Form
LE	<= or =< or ¬> or !>
LT	<

ternary-operator: EQ, NE, or SPANS
compares an item with a range of values. Ternary operators require a low value and a high value. You can also specify SPAN or SPANNING, and you can use these symbols:

Table 7.3

Operator	Alternate Form
EQ	=
NE	¬= or !=

value
is a literal value or the SYSTEM 2000 system string *TODAY*. Optionally, you can enclose a value with a delimiter of your choice. Sometimes you may need delimiters around character values, for example, to preserve a mixed case value. Any special character that appears at the beginning and end of a character value is assumed to be a delimiter. Consider these examples:

```
where c1 = 'Abc De' looks for Abc De
where c1 = @Abc De@ looks for Abc De
where c1 = @Abc De  looks for @Abc De
```

CONTAINS
searches for characters within an item's values.You can also specify CONT, CONTAIN, or CONTAINING.

text
For the syntax and explanation of CONTAINS text, see *SYSTEM 2000 QUEST Language.*

Examples

This section gives examples using different forms of the SYSTEM 2000 where-clause.

Unary operators

Unary operators search for values that exist or do not exist using the EXISTS and FAILS operators. The following where-clause qualifies data records having a value for the item ACCRUED VACATION.

```
where accrued vacation exists
```

The following where-clause qualifies data records not having a value for the item ACCRUED VACATION, that is, null items.

```
where accrued vacation fails
```

Note that SYSTEM 2000 unary operators are similar to SAS missing values expressions.

Binary operators

Binary operators compare items with a value or compare two items using the EQ, NE, GT, GE, LT, or LE operators (or their equivalent symbols). The following where-clause qualifies data records having the value for EMPLOYEE NUMBER equal to 1224.

```
where employee number=1224
```

The next where-clause qualifies data records where EMPLOYEE STATUS is not equal to FULL TIME. (It does not, however, qualify those records where EMPLOYEE STATUS is null as FAILS would.)

```
where employee status ne full time
```

The next where-clause qualifies data records where the value for HIRE DATE is greater than or equal to June 1, 1987.

```
where hire date=>06/01/1987
```

The next where-clause qualifies data records where the value for C105 equals the value for C4.

```
where C4 * EQ C105 *
```

Ternary operators

Ternary operators search for values in a range of values using the SPANS, EQ, and NE operators (or their equivalent symbols). The following where-clause qualifies data records where BIRTHDAY spans the dates January 1, l949 and January 31, 1949, inclusively.

```
wh birthday spans 01/01/1949 * 01/31/1949
```

CONTAINS operator

The CONTAINS operator searches for values that contain patterns of characters within values. The item must be a CHARACTER, TEXT, or UNDEFINED item. For example, the following where-clause qualifies data records where the values for STREET ADDRESS contain the character string RIM ROCK.

```
wh street address contains /RIM ROCK/
```

Combining conditions with AND and OR

Using the AND and OR operators, you can combine two or more conditions. AND combines two conditions by selecting values that satisfy both conditions, and OR combines two conditions by selecting values that satisfy either or both conditions. For example, the following where-clause qualifies data records having COBOL in the item SKILL TYPE and 4 in the item YEARS OF EXPERIENCE.

```
where skill type=cobol & years of experience=4
```

Not qualifying a condition with NOT

Using the NOT operator, you can select data records where values do not match a condition. For example, the following where-clause selects data records for the item PAY SCHEDULE that do not equal the value HOURLY or that are null.

```
wh ¬pay schedule=hourly
```

Designating specific types of records with HAS

Using the HAS operator, you can specify a focal record. For example in the following where-clause, the HAS operators specify C0 (the ENTRY record) as the focal record, because both conditions refer to the same schema record (C201). In this case, the HAS operators qualify C0 records that have the values COBOL and FORTRAN for C201. (If the HAS operator were not used, no records would qualify, because there would never be a C201 value of both COBOL and FORTRAN.)

```
wh C0 has c201 eq cobol and C0 has c201 eq fortran
```

Specifying position with AT

Using the AT operator, you can select values that are stored in a specified position in the database. Values must satisfy the condition and occupy a specific position. A data record's position is its number in a left-to-right enumeration below its parent record. For example, the following where-clause qualifies the data record in position 2 in a logical entry.

```
wh position title eq programmer at 2
```

Processing order

The order in which SYSTEM 2000 software processes conditions can affect which data records are selected. The software processes conditions with operators in this order: AT, HAS, NOT, AND, and OR.

When conditions are joined by the same operator, SYSTEM 2000 software first processes key conditions (ones that are indexed) from right to left, then non-key conditions (ones not indexed) from right to left.

You can alter processing order by changing the order of the conditions and by using parentheses around conditions. The software processes conditions enclosed in parentheses first.

For example, because the software processes the AND operator prior to the OR operator, to access those employees with an MBA degree and either a major or minor in Marketing, the following where-clause would yield the desired results:

```
wh degree=mba &
 (major field=marketing|minor field=marketing)
```

On the other hand, if you use the following where-clause, SYSTEM 2000 software would also select those employees who have a minor in Marketing and degrees other than MBAs.

```
wh degree=mba &
 major field=marketing|minor field=marketing
```

SYSTEM 2000 Ordering-clause

When you define a view descriptor, you can also include a SYSTEM 2000 ordering-clause to specify data order. You can reference only the items selected for the view descriptor. Without an ordering-clause or a SAS BY statement, the data order is determined by SYSTEM 2000 software.

A SAS BY statement automatically issues an ordering-clause to SYSTEM 2000 software. If a view descriptor already contains an ordering-clause, the BY statement overrides the ordering-clause for that program. An exception is when the SAS

procedure includes the NOTSORTED option. Then, the SAS BY statement is ignored, and the view descriptor ordering-clause is used.

Note: When you include a SYSTEM 2000 ordering-clause in a view descriptor, you can specify a terminator (either a colon or a semicolon). But if you specify both a where-clause and an ordering-clause, do not use a terminator between them. △

Syntax

ORDERED BY *sortkeys*;

ORDERED BY
is the keyword designating an ordering-clause. You can also use ORDER BY, OB, and SORT.

sortkeys
is a component name, component number, or SAS variable name of a SYSTEM 2000 item included in the view descriptor. Use commas to separate sort keys. You can also specify either ascending or descending order for each sort key.

ASCENDING|ASCEND|ASC|LOW|LO
specifies that you want the data ordered by ascending values of the sortkey. Ascending is the default.

DESCENDING|DESCEND|DESC|HIGH|HI
specifies that you want the data ordered by descending values of the sortkey.

If you specify more than one SYSTEM 2000 component, the values are ordered by the first named component, then the second, and so on. See *SYSTEM 2000 QUEST Language* for more details on the ordering-clause.

Example

The following ordering-clause causes the values to be presented in ascending order based on the values in item DEPARTMENT, then within departments in descending order based on the values in item SALARY:

```
order by department, desc salary
```

Creating and Using View Descriptors Efficiently

Follow these guidelines to minimize the use of SYSTEM 2000 software and system resources and to reduce the time SYSTEM 2000 software takes to access data.

- Select only the items your program needs. Selecting unnecessary items adds extra processing time.
- Use an ordering-clause or a SAS BY statement to specify the order in which logical entries are presented to the SAS System only if the SAS System needs the data in a particular order for subsequent processing. (The SAS BY statement issues an ordering-clause to SYSTEM 2000 software and overrides any existing ordering-clause for the view descriptor.) If you decide to use an ordering-clause or a SAS BY statement, order by an indexed item when possible.

  As an alternative to using an ordering-clause, which consumes CPU time each time you access the SYSTEM 2000 database, you could use the SORT procedure

with the OUT= option to create a sorted SAS data file. This is a better approach for data you want to use many times.

- If a view descriptor describes a large SYSTEM 2000 database and you will use the view descriptor often, it may be more efficient to extract the data and place them in a SAS data file. (Of course, the extracted data file will be very large but only created once. Also, the extracted data will not reflect any subsequent updates to the database.)
- When possible, specify selection criteria to subset the number of logical entries SYSTEM 2000 software returns to the SAS System.
- Write selection criteria that allow SYSTEM 2000 software to use available indexes when possible. This applies whether you specify the selection criteria as part of the view descriptor or use a SAS WHERE clause.

  This is especially important when accessing large databases. When SYSTEM 2000 software cannot use an index, it sequentially scans the entire database.

  You cannot guarantee that SYSTEM 2000 software will use an index to process a condition on a key item, but you can write selection criteria that allow SYSTEM 2000 software to use available indexes effectively. See *SYSTEM 2000 QUEST Language and System Commands* for a complete set of where-clause optimization guidelines.

ACCESS Procedure Data Conversions

The following table shows the default SAS System variable formats and informats that are assigned by the ACCESS procedure to each SYSTEM 2000 item type.

Table 7.4 Default SAS System Variable Formats and Informats for SYSTEM 2000 Item Types

SYSTEM 2000 Item Type and Picture	SAS Format and Informat
CHAR X(n)	$*n*
TEXT X(n)	$CHAR*n*.
DATE	DATE7.
INTEGER 9(n)	*n*.
DECIMAL 9(n).9(d)	*n*+d+1.*d*
MONEY 9(n).9(d)	*n*+d+1.*d*
REAL	BEST12.
DOUBLE	BEST12.
UNDEFINED X(n)	$HEXn*2.

If SYSTEM 2000 data fall outside valid SAS data ranges, you get an error message in the SAS LOG when you try to read the data. For example, a SYSTEM 2000 date may not fall in the valid SAS date range.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Interface to SYSTEM 2000® Data Management Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Interface to SYSTEM 2000® Data Management Software: Reference, Version 8

ISBN 1–58025–549–3

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

1st printing, October 1999