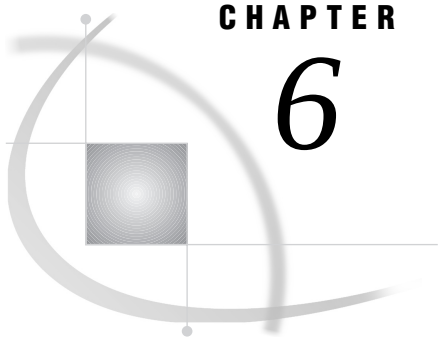

CHAPTER

6

ACCESS Procedure Reference

Introduction

The ACCESS procedure enables you to create and edit the descriptor files used by the SAS/ACCESS interface to CA-DATACOM/DB. This chapter provides reference information for the ACCESS procedure statements, including procedure syntax and statement options.

Additionally, the following sections provide information to help you optimize use of the interface:

- "Creating and Using View Descriptors Efficiently" on page 98 presents several efficiency considerations for using the SAS/ACCESS interface to CA-DATACOM/DB.
- "ACCESS Procedure Data Conversions" on page 99 summarizes how the SAS/ACCESS interface converts each type of CA-DATACOM/DB data into its SAS column format and informat equivalents.

For examples of how to use PROC ACCESS, refer to Chapter 3, "Defining SAS/ACCESS Descriptor Files," on page 15. If you need help with SAS data sets and data libraries, their naming conventions, or any terms used in the ACCESS procedure, refer to the *SAS Language Reference: Dictionary* and the *SAS Companion for the OS/390 Environment*.

Remember that help is available from within the ACCESS procedure by issuing the HELP command on any command line.

ACCESS Procedure Syntax

PROC ACCESS <*options*>;

Creating and Updating Statements

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

UPDATE *libref.member-name*.ACCESS|VIEW <*password-level=SAS-password*>;

Database-Description Statements

DATABASE | DB<=> <">*Datacom-database-name*<">;

DBSTAT<=> <">PROD<"> | <">TEST<"> | <">*test-version*<">;

PASSWORD | PASS | PW<=> <">*Datacom-password*<">;

TABLE<=> <">*Datacom-table-name*<">;

TBLSTAT<=> <">PROD<"> | <">TEST<"> | <">*test-version*<">;

URT<=> <">*User-Requirements-Table-name*<">;

USER<=> <">*authorized-Datacom-userid*<">;

Editing Statements

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

CONTENT <">*column-identifier-1*<"> <=> *SAS-date-format* | *length*
<...<">*column-identifier-n*<"> <=> *SAS-date-format* | *length*>;

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

EXTEND ALL | VIEW | <">*column-identifier-1*<">
<...<">*column-identifier-n*<">>;

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

INFORMAT | INFMT <">*column-identifier-1*<"> <=> *SAS-format-name*
<...<">*column-identifier-n*<"> <=> *SAS-format-name*>;

KEY<=> <">*Datacom-short-name*<">;

LIST ALL | VIEW | <">*column-identifier-1*<"> <...<">*column-identifier-n*<">;>

LISTINFO ALL | VIEW | <">*column-identifier-1*<">
<...<">*column-identifier-n*<">>;

LISTOCC <">*column-identifier-1*<"> <...<">*column-identifier-n*<">>;
OCCURS <">*column-identifier*<">
CONTENT *occurrence-1* <=> *SAS-format-name*
<...*occurrence-n* <=> *SAS-format-name*>;
|
DROP *occurrence-1* <TO> *occurrence-n*;
|
FORMAT <">*occurrence-1*<"> <=> *SAS-format-name*
<...<">*occurrence-n*<"> <=> *SAS-format-name*>;
|
INFORMAT <">*occurrence-1*<"> <=>*SAS-format-name*
<...<">*occurrence-n*<"> <=> *SAS-format-name*>;
|
RENAME <">*occurrence-1*<"> <=> *SAS-name*
<...<">*occurrence-n*<"> <=> *SAS-name*>;
|
RESET *occurrence-1* <TO> *occurrence-n*;
|
SELECT *occurrence-1* <TO> *occurrence-n*;
RENAME <">*column-identifier-1*<"> <=> *SAS-name*
<...<">*column-identifier-n*<"> <=>*SAS-name*>;
RESET ALL | <">*column-identifier-1*<"> <...<">*column-identifier-n*<">>;
SELECT ALL | <">*column-identifier-1*<"> <...<">*column-identifier-n*<">>;
SUBSET *selection-criteria*;
QUIT | EXIT;

Description

You use the ACCESS procedure to create and edit access descriptors and view descriptors, and to create SAS data files. Descriptor files describe DBMS data so that you can read, update, or extract the DBMS data directly from within a SAS session or in a SAS program.

The ACCESS procedure can run in batch or interactive line modes.

The following sections provide complete information on PROC ACCESS options and statements.

PROC ACCESS Statement Options

The ACCESS procedure statement takes the following options:

PROC ACCESS *options*;

Depending on which options you use, the ACCESS procedure statement performs several tasks.

You use the PROC ACCESS statement with database-description statements and certain procedure statements to create descriptors or SAS data files from DBMS data. See “Procedure Statements” on page 68 for information on which procedure statements to use for each task. The following sections describe PROC ACCESS options in greater detail.

Options

This section describes the options that you use to create and edit access descriptors and view descriptors.

ACCDESC=*libref.access-descriptor*
specifies an access descriptor. ACCDESC= is used with the DBMS= option to create a view descriptor that is based on the specified access descriptor. You specify the view descriptor's name in the CREATE statement. You can also use a SAS data set option on the ACCDESC= option to specify any passwords that have been assigned to the access descriptor.

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

DBMS=*DATACOM*
specifies the database management system you want to the descriptor(s) to access. Specify DBMS=DATACOM since you are using the SAS/ACCESS interface to CA-DATACOM/DB.

OUT=<*libref.*>*member-name*
specifies the SAS data file to which DBMS data are written. OUT= is used only with the VIEWDESC= option.

VIEWDESC=<*libref.*>*view-descriptor*
specifies a view-descriptor that accesses the CA-DATACOM/DB data. VIEWDESC= is used only with the OUT= option.

For example:

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

The VIEWDESC= option has two aliases: VD= and VIEW=.

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.

Table 6.1 on page 66 summarizes the levels of protection that SAS System passwords have and their effects on access descriptors and view descriptors.

Table 6.1 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 on the levels of protection and the types of passwords you can use, refer to the *SAS Language Reference: Dictionary*. The following section describes how you assign SAS System passwords to descriptors.

Assigning Passwords

You can assign a SAS password when you define a descriptor in the ACCESS procedure or after the descriptor file has been created by using PROC DATASETS.

Four password levels are available: READ=, WRITE=, ALTER=, or PW=. PW= assigns read, write, and alter privileges to a descriptor.

You can assign multiple levels of protection to a descriptor. However, for more than one level of protection (that is, both READ and ALTER), be sure to use a different password for each level. If you use the same password for each level, a user to whom you grant READ privileges only (in order to read the DBMS data) would also have privileges to alter your descriptor (which you do not want to allow).

ACCESS Procedure Method

To assign a password in the ACCESS procedure, specify the password level and password as a data set option in the CREATE statement. The following example creates and assigns passwords to an access descriptor and a view descriptor in the same procedure execution.

```
proc access dbms=Datacom;
  create work.emps.access (alter=rouge);
  table=employees;
  user=demo;

  create work.emp.view (alter=ego);
  select 1 2 3 4;
run;
```

Users will have to specify the ALTER password EGO to browse or edit the view descriptor and the ALTER password ROUGE to browse, edit, or define additional view descriptors from this access descriptor.

When creating a view descriptor from a password-protected access descriptor, specify the access descriptor password as a data set option after the ACCDESC= option. The following example specifies two data set options. The first specifies the access descriptor password and the second assigns a password to the view descriptor.

```
proc access dbms=Datacom ad=work.emps.access (alter=rouge);
  create work.emp2.view (alter=dumb);
  select 5 6 7 8;
run;
```

DATASETS Procedure Method

You assign a SAS password to an existing descriptor by using the DATASETS procedure. The DATASETS procedure MODIFY statement allows you to assign, change, and delete SAS passwords.

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=. 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 MYLIB.EMPLOYEE.

```
proc datasets library=mylib memtype=access;
   modify employee (alter=money);
run;
```

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

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

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

In this case, users are prompted for the SAS password when they try to read or update the DBMS data, or try to browse or edit the view descriptor VLIB.CUSPHON itself. You need both levels to protect the data and descriptor. Assign a WRITE level of protection to prevent data updates.

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

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

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

Procedure Statements

To invoke the ACCESS procedure you use the options described in “PROC ACCESS Statement Options” on page 65 and certain procedure statements. The options and statements that you choose are determined by your task.

□ To create an access descriptor:

PROC ACCESS DBMS=*DATACOM*;
  CREATE *libref.member-name*.ACCESS;
    database-description statements;
    optional editing statements;

RUN;

□ To create an access descriptor and a view descriptor:

PROC ACCESS DBMS=*DATACOM*;
  CREATE *libref.member-name*.ACCESS;
    database-description statements;
    optional editing statements;

CREATE *libref.member-name*.VIEW;
SELECT *item-list;*
optional editing statements;

RUN;

□ To create a view descriptor from an existing access descriptor:

PROC ACCESS DBMS=*DATACOM* ACCDESC=*libref.access-descriptor*;
CREATE *libref.member-name*.VIEW;
SELECT *item-list;*
optional editing statements;

RUN;

□ To update an access descriptor:

PROC ACCESS DBMS=*DATACOM*;
UPDATE *libref.member-name*.ACCESS;
procedure statements;

RUN;

□ To update a view descriptor:

PROC ACCESS DBMS=*DATACOM*;
UPDATE *libref.member-name*.VIEW;
procedure statements;

RUN;

CAUTION:
Updating access descriptors does not automatically update view descriptors. When you update an access descriptor (for example, drop another field from the display), the view descriptors based on this access descriptor are not updated automatically. You must re-create or modify any view descriptors that you want to reflect the changes made to the access descriptor. The view descriptors would still be valid, but they would no longer match the access descriptor. However, in some situations the view descriptors would no longer be valid (for example, if you re-create an access descriptor with the same name but base it on a different CA-DATACOM/DB table). △

CAUTION:
Altering CA-DATACOM/DB tables can affect descriptor files. Altering a CA-DATACOM/DB table that has descriptor files defined on it may cause these descriptors to be out-of-date or invalid. For example, if you add a field to a table and an existing access descriptor is defined on that table, the access descriptor does not reflect the new field, but it remains valid. However, if you delete a field or delete a table on which the view descriptor is based, the view descriptor fails when executed. Therefore, you must change the descriptor files manually when changes to CA-DATADICTIONARY invalidate them.

1 When you change CA-DATADICTIONARY, you must recreate the access descriptor(s) with PROC ACCESS, using the same name(s).

2 Then you must edit each view descriptor with PROC ACCESS. You will get a message if the view descriptor differs from its access descriptor. Change the view descriptor as needed.

△

The SAS/ACCESS interface view engine does a rudimentary validation of a view descriptor upon opening it. For example, the engine checks the data type information. If a problem is found, the engine writes a message to the log and stops.

For more information on the effects of changing a CA-DATACOM/DB table on existing view descriptors, see Appendix 1, "Information for the Database Administrator," on page 105.

Dictionary

ASSIGN

Specifies whether view descriptors created from an access descriptor will inherit or select their own SAS column names and formats.

Optional statement

Applies to: access descriptor

Syntax

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

Details The ASSIGN statement specifies whether view descriptors will inherit the SAS column names and formats assigned in the parent access descriptor at the time that the access descriptor was created, or whether the column names and formats can be selected in the view descriptor.

If you specify ASSIGN=YES, then default SAS column names and formats are generated for all CA-DATACOM/DB field names and these names and formats will be used in all derived view descriptors. You can edit the default column names and formats in the access descriptor with the RENAME, FORMAT, INFORMAT, and CONTENT statements, but you cannot edit them in the view descriptor.

If ASSIGN=NO, the default value, default names are not generated and any SAS column names assigned in the access descriptor can be edited in the view descriptor. If you do not specify any column names in the access descriptor, then fields selected in the view descriptor will use default SAS column names and formats, unless you edit them with the RENAME, FORMAT, INFORMAT, and CONTENT statements.

Default SAS column names follow these rules:

- If the CA-DATACOM/DB field name is longer than eight characters, the SAS System uses only the first eight characters. If truncating would result in duplicate names, numbers are appended to the end of the name. For example, the CA-DATACOM/DB field names CUSTOMERNAME and CUSTOMERNUMBER would become the SAS column names CUSTOMER and CUSTOME1.

- If the CA-DATACOM/DB field name contains invalid SAS name characters, such as a hyphen (-), the SAS System replaces them with underscores (_). For example, the CA-DATACOM/DB field name FUNC-INT becomes the SAS name FUNC_INT.
- For a key, the five-character DATACOM-NAME is used, if there is one. If that is missing, the first eight nonblank characters of the entity-occurrence name are used.
- For repeating fields, the system generates unique SAS names by suffixing or overlaying the occurrence number on the last position(s) of the SAS name. For example, the third occurrence of PHONE is PHONE3, the ninth occurrence of LASTNAME is LASTNAM9, and the eleventh occurrence of ADDRESS is ADDRES11. In some cases, this feature causes different fields to have SAS names that differ only in the suffixed number. For example, if you select BRANCH-NUMBER, BRANCH-PHONE, and BRANCH-ADDRESS and each repeats four times, the SAS names generated by default would be: BRANCH_1, BRANCH_2, BRANCH_3, BRANCH_4, BRANCH_5, BRANCH_6, BRANCH_7, BRANCH_8, BRANCH_9, BRANCH10, BRANCH11, and BRANCH12.

  The generated names are not listed in the LIST statement output.
- The SAS name for a compound field contains *GROUP*. To see the fully expanded repeating structure, use the LISTOCC statement. To see the field composition, use the LISTINFO statement.
- You can set different default names with a user exit, which is described in Appendix 2, “Advanced User Topics,” on page 117.

CONTENT

Specifies a SAS date format or length.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

CONTENT <">*column-identifier-1*<"> <=> *SAS-date-format* | *length*
<...<">*column-identifier-n*<"> <=> *SAS-date-format* | *length*>;

Details The CONTENT statement enables you to specify a SAS date format or column length. A date format means that the CA-DATACOM/DB data have the specified representation. The column length determines the number of characters to be accessed. The SAS System stores datetime values as the number of days and seconds before and after Jan. 1, 1960. Entering a SAS date format or column length automatically changes the default values for SAS formats and informats. However, if you have previously changed any format or informat values, specifying a CONTENT value does not change those values.

CA-DATACOM/DB does not have a date type; therefore, the CONTENT statement enables you to specify a date format for SAS processing. Four date formats are allowed:

- YYMMDD*w*. where *w* is 6 for two-digit years or 8 for four-digit years
- MMDDYY*w*. where *w* is 6 for two-digit years or 8 for four-digit years
- DDMMYY*w*. where *w* is 6 for two-digit years or 8 for four-digit years

□ JULIAN*w*. where *w* is 5 for two-digit years or 7 for four-digit years.

The *column-identifier* argument can be either the CA-DATACOM/DB field name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the access descriptor. If the column contains special characters or national characters, enclose the name in quotes.

If you specified ASSIGN=YES when creating an access descriptor, you cannot change the value for this statement when you later create a view descriptor based on that access descriptor.

You do not have to issue a SELECT statement for DBMS columns named in the CONTENT statement.

CREATE

Creates a SAS/ACCESS descriptor file.

Required statement

Applies to: access descriptor or view descriptor

Syntax

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

Details The CREATE statement identifies the access descriptor or view descriptor that you want to create. This statement is required for creating a descriptor file.

To create a descriptor, use a three-level name. The first level identifies the libref of the SAS data library where you will store the descriptor. You can store the descriptor in a temporary (WORK) or permanent SAS data library. The second level is the descriptor's name. The third level is the type of SAS file: 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 and view descriptors based on those 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.

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 below:

1. The CREATE statement for the access descriptor must directly follow the PROC ACCESS statement.
2. Database-description statements must follow the CREATE statement: TABLE, TBLSTAT, USER, PASSWORD, DATABASE, DBSTAT, and URT. The order of the database-description statements does not matter.
3. The editing statements must follow the database-description statements: ASSIGN, CONTENT, DROP, EXTEND, FORMAT, INFORMAT, KEY, LIST, LISTINFO, LISTOCC, OCCURS, QUIT, RENAME, and RESET. The SELECT and SUBSET statements are used only when creating view descriptors. QUIT is an editing statement but it terminates PROC ACCESS without creating your descriptor.

4 The RUN statement is used to signal the end of the ACCESS procedure.

Information from database-description statements is stored in the access descriptor; therefore, you do not need to repeat this information when you create view descriptors. However, if no security values were entered in the access descriptor, then you can use the database-description statements in a view descriptor to supply them.

View descriptors When you create a view descriptor for an existing access descriptor, you must use the ACCDESC= option with the ACCESS procedure.

When you create view descriptors and access descriptors in the same procedure execution, you must place the statements or groups of statements in the following order:

1 You must create an access descriptor before creating a view descriptor based on that access descriptor.
2 You should omit the RUN statement from the access descriptor specification.
3 Any database-description statements, such as PASSWORD, must precede the editing statements.
4 Among the editing statements, RENAME, CONTENT, FORMAT, and INFORMAT, can be specified only when ASSIGN=NO is specified in the access descriptor referenced by the view descriptor. The order of the statements within this group usually do not matter; see the individual statement descriptions for any restrictions.
5 The RUN statement is used to signal the end of the ACCESS procedure.

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, each descriptor is saved when a new descriptor is created or when the RUN statement is processed. If errors are found, error messages are written to the SAS log and processing is terminated. After you correct the errors, resubmit your statements.

For examples of how to create access descriptors and view descriptors, see Chapter 3, “Defining SAS/ACCESS Descriptor Files,” on page 15.

DATABASE

Identifies the CA-DATACOM/DB database to use.

Optional statement

Applies to: access descriptor

Syntax

DATABASE | DB<=> <">*Datacom-database-name*<">;

Details The DATABASE statement allows you to specify the name of the CA-DATACOM/DB database that contains the CA-DATACOM/DB table you want to access. In CA-DATACOM/DB, Database is a 32-character field that names an entity-occurrence of type DATABASE in CA-DATADICTIONARY.

The database name is required only if the table specified in the TABLE statement is not unique in the dictionary. If the name contains special characters or national characters, enclose it in quotes.

DB is the alias for the DATABASE statement.

DBSTAT

Indicates the status or version of the specified CA-DATACOM/DB database.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

DBSTAT<=> <">PROD<"> | <">TEST<"> | <">*test-version*<">;

Details The DBSTAT statement allows you to indicate the CA-DATADICTIONARY status and version of the CA-DATACOM/DB database that you want to access. The DBSTAT statement is required only if the database you want to use is not the one in production status.

The DBSTAT statement can take one of the following arguments:

PROD	indicates the database that is currently in production. This is the default.
TEST	indicates the database that is currently in test.
T plus a 3–digit number	indicates a specific test version of the database.

Other status values, such as HIST, are not allowed.

DROP

Drops a DBMS column so that it cannot be selected in a view descriptor.

Optional statement

Applies to: access descriptor

Syntax

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

Details The DROP statement drops the specified DBMS column from an access descriptor. The column therefore cannot be selected by a view descriptor that is based

on the access descriptor. However, the specified column in the DBMS table remains unaffected by this statement.

The *column-identifier* argument can be either the CA-DATACOM/DB field 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 drop the third and fifth columns, submit the following statement:

```
drop 3 5;
```

If the column name contains 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 the column name in the RESET statement. However, doing so also resets all of the column's attributes (such as SAS column name, format, and so on) to their default values.

EXTEND

Lists columns in the descriptor and gives information about them.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

EXTEND ALL | VIEW | <">*column-identifier-1*<"> <...<">*column-identifier-n*<">>;

Details The EXTEND statement lists information about the DBMS columns in a descriptor. The word *GROUP* is displayed to indicate the existence of a group.

You can use the EXTEND statement when creating an access or a view descriptor. The EXTEND information is written to your SAS log.

You can specify EXTEND as many times as you wish while creating a descriptor; specify EXTEND last in your PROC ACCESS code to see the completed descriptor information. Or, if you are creating multiple descriptors, specify EXTEND before the next CREATE statement to list all the information about the descriptor you are creating.

The EXTEND statement can take one of the following arguments:

ALL
: lists all of the DBMS columns in the file, the positional equivalents, the SAS names, the SAS informats, the database contents, the number of occurrences, and the DBMS column types (Alpha or Zoned). 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 DBMS columns that are selected for the view descriptor, along with their positional equivalents, their SAS names and informats, the database contents, number of occurrences, DBMS column types, any subsetting clauses, and the word ***SELECTED***. Any DBMS columns that are dropped in the access descriptor are not displayed. The VIEW argument is valid only for a view descriptor.

column-identifier
lists the specified DBMS column's SAS name, its positional equivalent, its SAS informat, the database content, number of occurrences, DBMS column type, and whether the column has been selected or dropped. If the column name contains special characters or national characters, enclose the name in quotes.

The *column-identifier* argument can be either the CA-DATACOM/DB field name, the positional equivalent from the LIST statement, which is the number that represents the column's place in the descriptor, or a list of names or positions. For example, to list information about the fifth column in the descriptor, submit the following statement:

```
extend 5;
```

Or, to list information about the fifth, sixth, and eighth columns in the descriptor, submit the following statement:

```
extend 5 6 8;
```

If the column contains special characters or national characters, enclose the name in quotes.

FORMAT

Changes the SAS format for a DBMS column.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

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

Details The FORMAT statement changes a SAS column format from its default format; the default SAS column format is based on the data type of the DBMS column. (See "ACCESS Procedure Data Conversions" on page 99 for information about the default formats that the ACCESS procedure assigns to your DBMS data types.)

The *column-identifier* argument can be either the CA-DATACOM/DB field 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 CA-DATACOM/DB field name contains 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: You do not have to issue a SELECT statement in a view descriptor for the columns included in the FORMAT statement. The FORMAT statement selects the columns. When you use the FORMAT statement in access descriptors, the FORMAT statement reselects columns that were previously dropped with the DROP statement. △

FMT is the alias for the FORMAT statement.

INFORMAT

Changes a SAS informat for a DBMS column.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

INFORMAT <">*column-identifier*<"> <=> *SAS-format-name*
<...<">*column-identifier-n*<"> <=> *SAS-format-name*>;

Details The INFORMAT statement changes a SAS column informat from its default informat; the default column informat is based on the data type of the DBMS column. (See "ACCESS Procedure Data Conversions" on page 99 for information about the default informats that the ACCESS procedure assigns to your DBMS data types.)

The *column-identifier* argument can be either the CA-DATACOM/DB field 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. informat with the BIRTHDATE column and with the second column in the access descriptor, submit the following statement:

```
informat 2=date9. birthdate=date9.;
```

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

You can use the INFORMAT 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: You do not have to issue a SELECT statement in a view descriptor for the columns included in the INFORMAT statement. The INFORMAT statement selects the columns. When you use the INFORMAT statement with access descriptors, the INFORMAT statement reselects columns that were previously dropped with the DROP statement. △

INFMT is the alias for the INFORMAT statement.

KEY

Specifies a key field that governs the order that records are read.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

KEY<=> <">*Datacom-short-name*<">;

Details The KEY statement specifies the CA-DATACOM/DB short name for a Default Key in the CA-DATACOM/DB table. The Default Key value governs the order in which records are read. The Default Key is an optional key that defaults to the Native Key. The Native Key governs how records are stored and maintained.

You can specify another key as the Default Key if you want the records processed in a certain order but you do not want to specify a SORT clause to achieve that result. You can also specify a Default Key with the DDBKEY= data set option when you execute a SAS procedure.

If the CA-DATACOM/DB short name contains special characters or national characters, enclose the name in quotes.

LIST

Lists columns in the descriptor and gives information about them.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

LIST ALL | VIEW |<">*column-identifier-1*<">
<... <">*column-identifier-n*<">>;

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

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 completed descriptor information. Or, if you are creating multiple descriptors, specify LIST before the next CREATE statement to list all the information about the descriptor you are creating.

The LIST statement can take one of the following arguments:

ALL
lists all the DBMS columns in the file, the positional equivalents, the SAS column names, and the SAS 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 DBMS columns that are selected for the view descriptor, along with their positional equivalents, their SAS column names and formats, any subsetting clauses, and the word ***SELECTED***. 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 DBMS column name, its positional equivalent, its SAS column name and format, and whether the column has been selected or dropped. If the column name contains special characters or national characters, enclose the name in quotes.

The *column-identifier* argument can be either the CA-DATACOM/DB field 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 list information about the fifth and eighth columns in the descriptor, submit the following statement:

```
list 5 8;
```

LISTINFO

Shows additional data field information.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

LISTINFO ALL | VIEW |<">*column-identifier-1*<">
<...<">*column-identifier-n*<">>;

Details The LISTINFO statement shows additional data field information for one or more DBMS columns in the descriptor. The LISTINFO statement can be used when creating an access or a view descriptor. The LISTINFO information is written to your SAS log.

The LISTINFO statement is especially helpful for key fields. It shows the CA-DATACOM/DB short name as well as all the columns and levels that make up the key.

The LISTINFO statement can take one of the following arguments:

ALL
lists the field composition of all the DBMS columns in the file.

VIEW
lists the field composition of the DBMS columns selected for the view descriptor. Any columns that are dropped in the access descriptor are not displayed. The VIEW argument is valid only for a view descriptor.

column-identifier
lists the field composition of the specified DBMS columns. If the column name contains special characters or national characters, enclose the name in quotes.

The *column-identifier* argument can be either the CA-DATACOM/DB field name, the positional equivalent from the LIST statement, which is the number that represents the column's place in the descriptor, or a list of names or positions. For example, to list information about the fifth column in the descriptor, submit the following statement:

```
listinfo 5;
```

Or, to list information about the fifth, sixth, and eighth columns in the descriptor, submit the following statement:

```
listinfo 5 6 8;
```

LISTOCC

Lists occurrences for repeating data fields.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

LISTOCC <">*column-identifier-1*<">
<... <">*column-identifier-n*<">>;

Details The LISTOCC statement lists all the occurrences for the specified repeating fields along with information such as the CA-DATACOM/DB level, the SAS column name, the occurrence number, the SAS column format and informat, the DB content, and whether the occurrence has been selected or dropped. The LISTOCC statement can be used when creating an access descriptor or a view descriptor. The LISTOCC information is written to your SAS log.

The LISTOCC statement takes the following argument:

column-identifier
The column-identifier argument can be either the CA-DATACOM/DB field 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 list occurrences for the fifth column in the descriptor, submit the following statement:

```
listocc 5;
```

If the DBMS column name contains special characters or national characters, enclose the name in quotes. The column-identifier must be a repeating field.

OCCURS

Modifies the occurrences of a repeating data field.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

OCCURS <">*column-identifier*<">
CONTENT <">*occurrence-1*<"> <=> *SAS-date-format | length*
<... <">*occurrence-n*<"><=> *SAS-date-format| length*>;
|
DROP *occurrence* <<TO> ... *occurrence-n*>;
|
FORMAT <">*occurrence-1*<"> <=> *SAS-format-name*
<... <">*occurrence-n*<"> <=> *SAS-format-name*>;
|
INFORMAT <">*occurrence-1*<"> <=> *SAS-format-name*
<... <">*occurrence-n*<"> <=> *SAS-format-name*>;
|
RENAME <">*occurrence-1*<"> <=> *SAS-name*
< ... <">*occurrence-n*<"> <=> *SAS-name*>;
|
RESET *occurrence-1* <<TO> ... *occurrence-n*>;
|
SELECT *occurrence* <<TO> ... *occurrence-n*>;

Details You use the OCCURS statement to modify values for occurrences of a repeating data field. The OCCURS statement can be used when creating an access descriptor or a view descriptor.

The OCCURS statement allows you to

- select individual occurrences or a range of occurrences
- drop individual occurrences or a range of occurrences
- reset individual occurrences or a range of occurrences
- change the format value for one or more occurrences
- change the informat value for one or more occurrences
- change the database content value for one or more occurrences
- rename the SAS column name for one or more occurrences.

You can see the two-character numeric level of a CA-DATACOM/DB field by using one of the LIST statements. The LVL column displays the word KEY for keys, the number 01 for simple fields and top-level compound fields, 02 for secondary fields, and so on. This listing is for information only and cannot be edited.

The column-identifier must be a repeating field, and can be the CA-DATACOM/DB field name or the positional equivalent from the LIST statement. The occurrence argument can be the occurrence name or the occurrence number. If the CA-DATACOM/DB field name or the occurrence name contains special characters, like `'-'`, enclose the name in quotes. The `'='`is optional for all subcommands.

You can use the LISTOCC statement to review your changes.

You do not have to issue a SELECT statement in a view descriptor for occurrences included in the CONTENT, FORMAT, INFORMAT, and RENAME subcommands. The subcommands select the columns.

The OCCURS statement can take one of the following subcommands:

SELECT
allows you to select individual occurrences to be included in your descriptor. This subcommand is used only when defining view descriptors.

You can select one or more individual occurrences or a range of occurrences. For example, if you want to select occurrences one, two, and three of the BRANCHOFFICE column in the CUSTOMERS table, submit the following statement:

```
occurs "BRANCHOFFICE" select 1 2 3;

                or

occurs "BRANCHOFFICE" select 1 to 3;
```

DROP
allows you to drop individual occurrences from your descriptor. If you drop all occurrences of a column, the column is automatically dropped. This subcommand is used only when defining access descriptors.

You can drop one or more individual occurrences or a range of occurrences. For example, if you want to drop occurrences one, two, and three of the BRANCHOFFICE column in the CUSTOMERS table, submit the following statement:

```
occurs "BRANCHOFFICE" drop 1 2 3;

                or

occurs "BRANCHOFFICE" drop 1 to 3;
```

RESET
allows you to reset the attributes of individual occurrences. This subcommand can be used when creating an access or view descriptor. Specifying the RESET subcommand for an occurrence has the same effect on occurrence attributes as specifying the RESET statement for a column. See “RESET” on page 85 for more information.

You can reset one or more individual occurrences or a range of occurrences. For example, if you want to reset occurrences one, two, and three of the BRANCHOFFICE column in the CUSTOMERS table, submit the following statement:

```
occurs "BRANCHOFFICE" reset 1 2 3;

                or

occurs "BRANCHOFFICE" reset 1 to 3;
```

FORMAT
allows you to change the format attribute of individual occurrences. This subcommand can be used when creating access or view descriptors. However, the format attribute cannot be changed in a view descriptor when you set ASSIGN=YES.

You can change the format attribute of one or more occurrences in one FORMAT subcommand. For example, if you want to change the format attribute for occurrences nine and ten of the BRANCHOFFICE column in the CUSTOMER table, submit the following statement:

```
occurs "BRANCHOFFICE" format 9 $21.  10 = $8.;
```

INFORMAT
allows you to change the informat attribute of an individual occurrence. This subcommand can be used when creating access or view descriptors. However, the informat attribute cannot be changed in a view descriptor when you set ASSIGN=YES.

You can change the informat attribute of one or more occurrences in one INFORMAT subcommand. For example, if the BRANCHOFFICE column in the CUSTOMERS table is a repeating field, and you want to change the informat attribute for occurrences nine and ten, submit the following statement:

```
occurs "BRANCHOFFICE" informat 9 $21. 10 = $8.;
```

CONTENT
allows you to change the DB content attribute of an individual occurrence. This subcommand can be used when creating access or view descriptors. Changing the DB content attribute of an occurrence has the same effect on the SAS formats and informats for CA-DATACOM/DB tables and records as changing the DB content attribute of a column. See “CONTENT” on page 71 for more information. For example, if the FIRSTORDERDATE column in the CUSTOMERS table is a repeating field, and you want to change the DB content attribute for occurrences nine and ten, submit the following statement:

```
occurs firstorderdate content 9 yymmdd6.  10 = yymmdd6.;
```

RENAME
allows you to rename a SAS column name for an individual occurrence. This subcommand can be used when creating an access or view descriptor. However, this subcommand has different effects on access and view descriptors based on the value specified in the ASSIGN statement.

If you set ASSIGN=NO in the access descriptor, the SAS column name can be renamed. If you set ASSIGN=YES, the SAS column name can be renamed in the access descriptor but not in the view descriptor.

You can rename the SAS column name for one or more occurrences in one RENAME subcommand. For example, if you want to rename occurrences nine and ten of the BRANCH-OFFICE column in the CUSTOMERS table, submit the following statement:

```
occurs "BRANCH-OFFICE" rename 9 london 10 = tokyo;
```

The result of selecting a key that consists of multiple fields is always a SAS character column. The value of the SAS column is the concatenated values of the component fields. If any of the component fields are numeric, they are converted to character representation, with a format and length set by the interface view engine. The character-only restriction exists so that the key can be used in a WHERE clause.

PASSWORD

Specifies a CA-DATADICTIONARY password.

Optional statement

Applies to: access descriptor

Syntax

PASSWORD | PASS | PW<=> <">*Datacom-password*<">;

Details The PASSWORD statement allows you to supply a CA-DATADICTIONARY password. Not every userid requires a password.

The value is the 12–character PASSWORD attribute of the PERSON entity-occurrence identified in User Name. If you enter a value, it is saved (in encrypted form) in the access descriptor and in any view descriptor created from it.

If the password contains any special characters or national characters, enclose it in quotes.

PASS and PW are aliases for the PASSWORD statement.

QUIT

Terminates the procedure.

Optional statement

Applies to: access descriptor or 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

Modifies the SAS column name.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

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

Details The RENAME statement enters or modifies the SAS column name that is associated with a DBMS column. The RENAME statement can be used when creating an access descriptor or a view descriptor. However, the value of the ASSIGN statement affects when the RENAME statement can be used.

When you create an access descriptor, the default setting for a SAS column name is a blank. When ASSIGN=YES, default SAS column names are generated and these SAS column names are used by all of the view descriptors derived from this access descriptor. You can use the RENAME statement to edit the SAS column names assigned in the access descriptor and these renamed SAS column will be used by its view descriptors, unless a RESET statement or another RENAME statement is used in the access descriptor.

If you omit the ASSIGN statement or specify it with a **NO** value, you can use the RENAME statement to assign a SAS column name. In this case, the SAS column names that you enter in the access descriptor will be retained by any view descriptors derived from this access descriptor; however, you can edit them in the view descriptor with the RENAME statement. Column names renamed in the view descriptor apply only to that view descriptor.

The *column-identifier* argument can be either the CA-DATACOM/DB field 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 column names that are associated with the seventh DBMS column and the nine-character FIRSTNAME DBMS column in a descriptor, submit the following statement:

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

The DBMS column name (or positional equivalent) is specified on the left side of the expression, with the SAS column name on the right side. The equal sign (=) is optional. If the CA-DATACOM/DB field name contains 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 column name associated with a DBMS column, you do not have to issue a SELECT statement for that column.

RESET

Resets DBMS columns to their default settings.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

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

Details The RESET statement resets either the attributes of all the DBMS columns or the attributes of the specified DBMS columns to their default values. The RESET statement can be used when creating an access descriptor or a view descriptor. However, this statement has different effects on access and view descriptors, as described below.

Access descriptors When you create an access descriptor, the default setting for a SAS column name is a blank. However, if you have previously entered or modified any of the SAS column names, the RESET statement resets the modified names to the default names that are generated by the ACCESS procedure. How the default SAS column 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 CA-DATACOM/DB field name.

The current SAS column format and informat are reset to the default SAS format and informat, which was determined from the DBMS column's data type. The current DB content is also reset to the default value. Any columns that were previously dropped, that are specified in the RESET command, 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 column and then select it again within the same procedure execution, the SAS column name, format, informat, and database content are reset to their default values (the SAS name is generated from the DBMS column name, and the format and informat values are generated from the data type). 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 cannot be used to restore the default column names and formats in the view descriptor, but it will affect the SELECT statement for the view.

The RESET statement can take one of the following arguments:

ALL
for access descriptors, resets all the DBMS 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.

column-identifier
can be either the CA-DATACOM/DB field 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 third column, submit the following statement:

```
reset 3;
```

If the CA-DATACOM/DB field name contains 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.

SELECT

Selects DBMS columns for the view descriptor.

Required statement

Applies to: view descriptor

Syntax

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

Details The SELECT statement specifies which DBMS columns in the access descriptor to include in the view descriptor. This is a required statement and is used only when defining view descriptors.

The SELECT statement can take one of the following arguments:

ALL
includes in the view descriptor all the DBMS columns that were defined in the access descriptor excluding dropped columns.

column-identifier
can be either the CA-DATACOM/DB field 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 CA-DATACOM/DB field name contains special characters or national characters, enclose the name in quotes. You can select as many DBMS 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 DBMS table.

The selection-criteria argument can be either a WHERE clause or a SORT clause. For more information on the WHERE clause, see "WHERE Clause in a View Descriptor" on page 91. For more information on the SORT clause, see "SORT Clause in a View Descriptor" on page 97. You can use either SAS column names or DBMS column names in your selection criteria. Specify your WHERE clause and SORT clause by using the same or separate SUBSET statements. For example, you can submit the following SUBSET statements:

```
subset where jobcode = 1204;
subset sort lastname;
subset where jobcode=1204 sort lastname;
```

The SAS System does not check the SUBSET statement for errors. The statement is verified and validated only when the view descriptor is used in a SAS program.

To delete the selection criteria, submit a SUBSET statement without any arguments.

TABLE

Indicates the CA-DATACOM/DB table you want to use.

Required statement

Applies to: access descriptor

Syntax

TABLE<=> <">*Datacom-table-name*<">;

Details The TABLE statement specifies the CA-DATACOM/DB table that you want to access. *Datacom-table-name* is the 32-character field that names an entity-occurrence of type RECORD in the CA-DATADICTIONARY. (For CA-DATACOM/DB R8, the type is TABLE.)

The TABLE statement is required to create an access descriptor and the table name must be unique. If the table name is not unique, you can combine the TABLE statement with the DATABASE, DBSTAT, and TBLSTAT statements until a unique combination is identified.

If the table name contains special characters or national characters, enclose the name in quotes.

TBLSTAT

Indicates the status or version of the specified CA-DATACOM/DB table.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

TBLSTAT<=> <">PROD<"> | <">TEST<"> | <">*test-version*<">;

Details The TBLSTAT statement allows you to indicate the CA-DATADICTIONARY status and version of the CA-DATACOM/DB table you want to access. The TBLSTAT statement is required only if the database you want to use is not the one in production status.

The TBLSTAT statement can take one of the following arguments:

PROD	indicates the table that is currently in production. This is the default.
TEST	indicates the table that is currently in test.
T plus a 3–digit number	indicates a specific test version of the table.

Other status values, such as HIST, are not allowed.

UPDATE

Updates a SAS/ACCESS descriptor file.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

UPDATE *libref.member-name*.ACCESS | VIEW
<*password-level=SAS-password*>;

Details The UPDATE statement identifies an existing access descriptor or view descriptor that you want to update (edit). The descriptor can exist in either a temporary (WORK) or permanent SAS data library. If the descriptor has been protected with a SAS password that prohibits editing of the ACCESS or VIEW descriptor, then the password must be specified on the UPDATE statement.

Note: It is recommended that you re-create (or overwrite) your descriptors rather than update them. SAS does not validate updated descriptors. If you create an error while updating a descriptor, you will not know of it until you use the descriptor in a SAS procedure such as PROC PRINT. △

To update a descriptor, use its three-level name. The first level identifies the libref of the SAS data library where you stored the descriptor. The second level is the descriptor's name (member name). The third level is the type of SAS file: ACCESS or VIEW.

You can use the UPDATE statement as many times as necessary in one procedure execution. That is, you can update 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 update access descriptors and view descriptors in separate executions of the procedure.

You can use the CREATE statement and the UPDATE statement in the same procedure execution.

If you update only one descriptor in a procedure execution, the UPDATE statement must be the first statement after the PROC ACCESS statement (Note: The ACCDESC= parameter cannot be specified on the PROC ACCESS statement).

The following statements are not supported when using the UPDATE statement: ASSIGN, RESET, SELECT, and OCCURS subcommands RESET and SELECT.

Note: You cannot create a view descriptor after you have updated a view descriptor in the same procedure execution. You can create a view descriptor after updating or creating an access descriptor or after creating a view descriptor. △

The following example updates the access descriptor MYLIB.ORDERS on the CA-DATACOM/DB table ORDER. In this example, the SAS column names are changed and formats are added.

```
proc access dbms=Datacom;
   update mylib.orders.access;
   rename ordernum ord_num
          fabriccharges fabrics;
   format firstorderdate date7.;
   informat firstorderdate date7.;
   content firstorderdate yymmdd6.;
 run;
```

The following example updates an access descriptor MYLIB.EMPLOYEE on the CA-DATACOM/DB table EMPLOYEES and then re-creates a view descriptor VLIB.EMPS, which was based on MYLIB.EMPLOYEE. The original access descriptor included all of the DBMS columns in the table. Here, the SALARY and BIRTHDATE columns are dropped from the access descriptor so that users cannot see this data. Because RESET is not supported when UPDATE is used, the view descriptor VLIB.EMPS must be re-created in order to omit the SALARY and BIRTHDATE columns.

```
proc access dbms=Datacom;
   /* update access descriptor  */
   update mylib.employee.access;
   drop salary birthdate;
   list all;

   /* re-create view descriptor */
   create vlib.emps.view;
   select empid hiredate dept jobcode sex
          lastname firstname middlename phone;
   format empid 6.
          hiredate date7.;
   subset where jobcode=1204;
run;
```

URT

Identifies the User Requirements Table to use.

Optional statement

Applies to: access descriptor or view descriptor

Syntax

URT<=> <">*User-Requirements-Table-name*<">;

Details The URT statement identifies the User Requirements Table (URT) to be used by the interface view engine when it opens a view descriptor. CA-DATACOM/DB requires a URT to open a table. For more information, see "User Requirements Table (URT)" on page 112. However, this statement is optional. If you do not specify a URT when creating an access descriptor or a view descriptor, the engine will create a default URT. A URT name can also be provided with a data set option (see "Data Set Options" on page 117).

If you specify a URT when creating an access descriptor, the interface view engine will use when it opens any view descriptors created from this access descriptor.

If the URT name contains special characters or national characters, enclose the name in quotes.

USER

Specifies a CA-DATADICTIONARY userid.

Required statement

Applies to: access descriptor

Syntax

USER<=> <">*authorized-Datacom-userid*<">;

Details The USER statement supplies a required CA-DATADICTIONARY userid. The userid is a 32–character entity-occurrence name of a PERSON entity in CA-DATADICTIONARY, which is not necessarily the same as the user's TSO id.

The value entered in the USER statement is saved in the access descriptor and in any view descriptor created from the access descriptor. The user name and optional password must have retrieval authority on six entity-types: DATABASE, FILE, RECORD, ELEMENT, KEY, and FIELD.

If the userid contains special characters or national characters, enclose it in quotes.

WHERE Clause in a View Descriptor

You can use a WHERE Clause in a view descriptor to select specific records from a CA-DATACOM/DB table. You can reference any CA-DATACOM/DB field included in the view descriptor.

View WHERE Clause Syntax

A WHERE clause in a view descriptor consists of the word WHERE followed by one or more conditions that specify criteria for selecting records from one CA-DATACOM/DB table. (WITH and WH are valid synonyms for the word WHERE.)

A condition can be one of the following:

```
field-name<(occurrence)>|key-name operator value
field-name* operator field-name*
field-name<(occurrence)>|key-name range-operator low-value * high-value
```

The user-supplied elements of the WHERE clause conditions are described here.

field-name<(occurrence)>|key-name
is the CA-DATACOM/DB name of the field or key for which you are specifying criteria. The field must be selected in the view descriptor. The interface view engine assumes that the name in a condition is a SAS name. If it is not, the name will be treated as a CA-DATACOM/DB name.

If the field is a repeating field, you must specify the occurrence of that field in parenthesis, where occurrence is one of the following:

n	indicates the nth occurrence. For example, `where address(3) contains dallas` selects those records where the third occurrence of ADDRESS contains DALLAS.
ALL	indicates all occurrences selected in the view descriptor. For example, the WHERE clause below selects those records where all occurrences of ADDRESS contains DALLAS. `where address(all) contains dallas`
ANY	indicates any occurrence. An asterisk (*) can be used instead of ANY. For example, `where address(any) contains dallas` selects those records where any occurrence of ADDRESS contain DALLAS. You could have used ADDRESS(*) instead.

operator
is one of the following:

= or EQ	equal to
> or GT	greater than
< or LT	less than
!= or ¬= or NE	not equal
>= or GE or GTE	greater than or equal to
<= or LE or LTE	less than or equal to
CONTAINS or CONTAINING	contains
¬ CONTAIN or ¬ CONTAINING	does not contain

!CONTAIN or !CONTAINING	does not contain

range-operator
is one of the following:

= or EQ or SPANS	is within the range (inclusive)
!= or ¬= or NE	is outside the range

value, *high-value*, and *low-value*
represent valid values for the field or key.

For more information, see "Specifying Values in WHERE Clauses" on page 94.

View WHERE Clause Examples

The asterisk (*) is required when comparing two field-names. For example, the folowing WHERE clause selects those records where the wages are less than the commission:

```
where ytd-wages*<ytd-commission*
```

This WHERE clause

```
where ship-quant*=order-quantity*
```

selects those records where the ship-quantity is equal to the order-quantity.

The asterisk is also required when comparing low and high range values. For example, the following WHERE clause selects employees with employee numbers between 2300 and 2400:

```
where number spans 2300*2400
```

The WHERE clause

```
where lastname spans 'A'*'Smith'
```

selects those employees with last names up to Smith. See "Character Fields" on page 94 for details on the use of quotes.

If the asterisk appears in a value, enclose the value in quotes or use the DDBSPANS system option to specify another special character. For more information on system options, see "System Options" on page 115.

Expressions

Conditions can be combined to form expressions. Two conditions can be joined with OR (|) or AND (&). Since expressions within parentheses are processed before those outside, use parentheses to have the OR processed before the AND.

```
where cost=.50 & (type=ansi12 | class=sorry)
```

Conditions can also be preceded with NOT (X).

```
where cost=.50 & not (type=ansi12 | class=sorry)
```

The following WHERE clause selects all records where AVAIL is Y or W:

```
where avail eq y | avail eq w
```

The next WHERE clause selects all records where PART is 9846 and ON-HAND is greater than $2x10^6$:

```
where part=9846 & on-hand>2.Oe+6
```

Specifying Values in WHERE Clauses

The next few pages discuss guidelines and considerations that govern how you specify values in WHERE clause conditions.

Character Fields

For character fields you can use quoted or unquoted strings. Any value entered within quotes is left as is; all unquoted values are uppercased, and redundant blanks are removed. For example,

```
where lastname=Smith
```

extracts data for SMITH, and the next example extracts data for Smith:

```
where lastname='Smith'
```

If the value is shorter than the field, it is padded on the right with blanks before the comparison. (No padding is done if you use the CONTAINS operator.) If the value is longer than the field, it is truncated to the field length before the comparison is done. The WHERE clause

```
where name=Anderson
```

selects all records where NAME is ANDERSON. The WHERE clause

```
where city='TRUTH OR CONSEQUENCES' | stzip='NM  87901'
```

selects all records where CITY is TRUTH OR CONSEQUENCES or STZIP is NM 87901. Notice in the first condition that quotes prevent OR from being used as an operator. In the second condition, they prevent the extra space between NM and 87901 from being removed.

In this example, either of these WHERE clauses

```
where shop='Joe''s Garage'
where shop=''Joe;s Garage''
```

selects all records where SHOP is Joe's Garage. Because the value is enclosed in quotes, the two consecutive single quotes are treated as one quote. You can also use double quotes around a value. Also, two consecutive double quotes become one double quote if surrounded by double quotes. If two consecutive double quotes are surrounded by single quotes, they remain two double quotes and vice versa.

Date Values

You can use the DB Content statement to specify a date format. Using this statement, you can specify the dates according to your SAS informat. Do not use 'd as you would for SAS software.

$HEX. Format Fields

For fields that are converted to $HEX. format because of their data type or length (see "ACCESS Procedure Data Conversions" on page 99), the value must be specified in hexadecimal. A value longer than the field is truncated to the field length before the comparison is done. A value shorter than the field is padded on the right with binary zeros before the comparison. For example, if CODE has $HEX4. format,

```
where code=f1f
```

extracts the data for CODE equals 10 (F1F0).

Values That Do Not Fit the Field Picture

If you specify a value that does not fit the field's picture, you may receive an error, or the value may be adjusted to fit the picture before sending the request to CA-DATACOM/DB.

The following examples illustrate how various misfit values are handled. Assume throughout that COST has a database length of 5, with 2 decimals.

In the first set of examples, some misfit values produce errors, some are truncated, and some cause operators to be changed. Errors occur when the equals operator or not equals operator is used with a misfit value. Operators are changed when that change plus truncation means the value will fit the picture and still produce the results you intended.

Table 6.2

Condition	Request Sent to CA-DATACOM/DB	
cost=.003	Error	(underflow: field has two decimals)
cost>.003	cost>0.00	(truncated)
cost>3.0052	cost>3	(truncated)
cost<.0001	cost ≤ 0.00	(truncated, < changed to ≤)
cost<20.001	cost ≤ 20	(truncated, < changed to ≤)

The next examples show values that exceed the field size. If possible, your values are replaced with the largest value that can be stored in the field.

Table 6.3

Condition	Request Sent to CA-DATACOM/DB
cost<11123	cost ≤ 999.99
cost ¬= 9999	Error (overflow, field cannot store integers > 999)
cost >= -12345	cost ≥ - 999.99

Masking Values

When a condition includes the EQ, NE, CONTAINS, or NOT CONTAINS operator and the field is in display code, you can mask the value. That is, you can specify that only certain positions within the value are to be compared to those positions in the field. A pound sign (#) marks the positions that you do not want to be compared. For example,

```
where zipcode eq 7#8
```

selects all records with zip codes that have a 7 in the first position and an 8 in the third position. The condition

```
where lastname contains m#n
```

selects all records with last names such as Mendoza, Harman, and Warminsky.

If you use the EQ or NE operators and you mask a value that is shorter than the database field, your values are padded on the right with mask characters. (No padding is done for NOT CONTAINS.) For example,

```
where lastname eq m#n
```

would select records with last names such as Mendoza, McNeal, and Monroe. Names such as Harman or Warminsky would not qualify.

Use the DDBMASK system option to change the default masking character (#). For more information on system options, see “System Options” on page 115.

Multi-Field Keys

For a condition that specifies a multi-field key, you may need to enclose each value with delimiters.

Note: You cannot use compound fields in the WHERE clause. △

For multi-key fields, use a delimiter character* before and after each value if the value you are entering is not the same length as the multi-field key and you are using either NOT CONTAINS or the mask character. Values for keys are always in display code. For example, suppose INIT-ID is a multi-key field. INIT is a character field of length 3, and ID is a numeric field of length 7. The WHERE clause

```
where init-id=\jde\27#\
```

selects all records where the initials are JDE and the ID number starts with 27. Your value for ID is padded on the right with mask characters, so the entire value is treated as if you had specified JDE27#####.

You can omit delimiters if you specify the same number of characters as the multi-field key contains. For example, this WHERE clause

```
where init-id=jde27#####
```

also selects all records where the initials are JDE and the ID number starts with 27, just as in the previous example. No delimiters are required here because JDE27##### is 10 characters long, which is the same size as the key field.

When you do not include delimiters or masked characters in the value, blanks or zeros are used for padding. The WHERE clause

```
where weight-sex=78m
```

selects all records where weight equals 78 and sex equals M. The value is treated as if it had been specified as \78\m\.

On the other hand, the WHERE clause

```
where age-degree=25bs
```

selects all records where age equals 25 and degree equals BS. The value is treated as if it had been specified as \25\bs \.

Note: A considerable amount of processing is required when a procedure must convert an apparently simple condition into a complex request to CA-DATACOM/DB. For example, if the fields AGE and SEX are not contiguous within the record, the procedure converts the condition AGE-SEX<25M to SEX<M OR (SEX=M AND AGE<25) before submitting the request. CA-DATACOM/DB, in turn, processes the request and, if possible, uses permanent indexes to satisfy it. △

Guidelines

Consider the following guidelines when you specify a WHERE clause in the view descriptor:

---

* Use the DDBDELIM system option to change the default delimiter character (\). For more information on system options, see “System Options” on page 115.

- You can enter a WHERE clause or a SORT clause or both, in either order. But if you enter both, do not use a terminator between them.
- The keyword WHERE is not required unless the WHERE clause is the second clause (following the SORT clause). The SORT clause must begin with SORT.
- CA-DATACOM/DB does not have a date data type. However, the selection criteria will honor a SAS date format if you specify one in the CONTENT and INFORMAT statements.
- The CA-DATACOM/DB fields must be selected in the view descriptor in order for you to use them in the WHERE clause.
- All conditions in the WHERE clause must refer to fields in a single table. To join conditions that pertain to two CA-DATACOM/DB tables, use the SQL procedure.
- If you enter a SAS WHERE clause when you use the view descriptor in a SAS procedure, the SAS WHERE clause is connected to the WHERE clause in the view descriptor (if any) with the AND operator.
- The WHERE clause is not parsed (or checked) until the interface view engine tries to execute it for a procedure.
- Field names in the WHERE clause conditions can be SAS names or CA-DATACOM/DB names. However, you should use SAS names for repeating fields or for fields within repeating fields.
- Character literals and values for zoned decimal fields can contain the pound sign (#) to indicate masking out characters for pattern matching operations.

For more information on specifying WHERE clauses, see “Deciding How to Specify Selection Criteria” on page 121.

SORT Clause in a View Descriptor

When you define a view descriptor, you can also include a SORT clause to specify data order. You can reference only the CA-DATACOM/DB fields selected for the view descriptor.

Without a SORT clause or a SAS BY statement, the data order is determined by the Native Key for the CA-DATACOM/DB table (or by the Default Key specified in the access or view descriptor).

A SAS BY statement automatically issues a SORT clause to CA-DATACOM/DB. However, the SAS BY statement may cause grouping of the output results in some procedures; this may not be what you want.

If a view descriptor already contains a SORT clause, the BY statement overrides the SORT 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 SORT clause is used.

View SORT Clause Syntax

The syntax for the SORT clause is

```
SORT field-name <ASCENDING|UP|A> <DESCENDING|DOWN|D>
   <,field-name...>
```

The elements of the SORT clause are described here.

field-name
is a CA-DATACOM/DB field name or SAS column name of a CA-DATACOM/DB field included in the view descriptor. Use commas to separate sort keys. You can also specify either ascending or descending order for each field name.

ASCENDING | UP | A
specifies that you want the data ordered by ascending values of the *field-name*. ASCENDING is the default.

DESCENDING | DOWN | D
specifies that you want the data ordered by descending values of the *field-name*.

If you specify more than one CA-DATACOM/DB field, the values are ordered by the first named field, then the second, and so on.

SORT Clause Example

The following SORT clause causes the values to be presented in ascending order based on the values in field STATE, then within states in descending order based on the values in field CITY:

```
sort state, city down
```

Guidelines

Consider the following guidelines when you specify a SORT clause in the view descriptor:

- You can enter a WHERE clause or a SORT clause or both, in either order. But if you enter both, do not use a terminator between them.
- The keyword WHERE is not required unless the WHERE clause is the second clause (following the SORT clause). The SORT clause must begin with SORT.
- If you specify a SAS BY clause when you execute a procedure, it replaces the SORT clause in the view descriptor. However, if the SAS procedure includes the NOTSORTED option, the SAS BY clause is ignored and the SORT clause in the view descriptor is used. A message is written to the LOG window when the NOTSORTED option causes a SORT clause to be ignored.
- The CA-DATACOM/DB fields must be selected in the view descriptor in order for you to use them in the SORT clause.
- In the SORT clause, you can specify multiple fields, separated by commas.
- The SORT clause is not parsed (or checked) until the interface view engine tries to execute it for a procedure.
- Field names in the SORT clause conditions can be SAS names or CA-DATACOM/DB names. However, you should use SAS names for repeating fields or for fields within repeating fields.

Creating and Using View Descriptors Efficiently

When creating or using view descriptors, follow these guidelines to minimize the use of CA-DATACOM/DB and your operating system resources and to reduce the time CA-DATACOM/DB takes to access data.

- Select only the fields your program needs. Selecting unnecessary fields adds extra processing time.

- Specify the order in which records are presented to the SAS System (with a SORT clause or a SAS BY statement) only if the SAS System needs the data in a particular order for subsequent processing.

  The SAS BY statement issues an ordering clause to CA-DATACOM/DB so that CA-DATACOM/DB does the sorting using its system resources. This SORT clause overrides any existing SORT clause for the view descriptor. If you decide to use a SORT clause or a SAS BY statement, order by a key, which is indexed, when possible. (For help in determining which fields in a table are indexed, see your DBA or the table's creator.)

  As an alternative to using a SORT clause, which consumes CPU time each time you access the CA-DATACOM/DB table, 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 CA-DATACOM/DB table 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. (Even though the extracted data file will be very large, you will need to create it only once. Also, the extracted data will not reflect any subsequent updates to the table.) See "Performance Considerations" on page 43 for more information on when it is best to extract data.
- Specify selection criteria to retrieve a subset of the records CA-DATACOM/DB software returns to the SAS System, where possible.
- If you use a Default Key, the interface view engine will use an index read instead of a sort if it can. Index reads are faster, but not always possible. For example, an index read is not possible if you specify multiple sort keys, multiple WHERE clause conditions, or a WHERE clause condition with a column that is not a key.
- Omit the KEY statement if you do not need a certain order and you want to retrieve the data sequentially. Otherwise, you may cause an unnecessary sort. PROC FSBROWSE, FSEDIT, and FSVIEW automatically use random access and require a value in the Default Key field.
- You can provide your own URT that is fine-tuned for your applications.

ACCESS Procedure Data Conversions

The following table shows the default formats that the SAS System assigns to each CA-DATACOM/DB data type. The default formats also become the default informats.

Table 6.4 Default SAS System Column Formats for CA-DATACOM/DB Data Types

Field Type	Field Description	Default SAS Format
C	character	$*len*.
B	binary:	
	for length ≤ 8, unsigned	(2x*len*+1).*dec**
	for length 8, signed	(2x*len*+2).*dec**
	for length > 8	$HEX(2x*len*).
D	packed decimal:	
	for length ≤16, unsigned	(2x*len*+1).*dec**

Field Type	Field Description	Default SAS Format
	for length 16, signed	(2x*len*+2).*dec**
	for length > 16	$HEX(2x*len*).
E	extended floating-point	$HEX(2x*len*).
G	graphics data	$HEX(2x*len*).
H	hex character	$*len*.
K	Kanji (same as Y)	$HEX(2x*len*).
L	long floating-point	E24.
N	numeric (zoned decimal):	
	for length 16, unsigned	*len.dec*
	for length 16, signed	(*len*+1).*dec*
	for length > 16	$HEX(2x*len*).
S	short floating-point	E14.
T	PL/I bit representation	$HEX(2x*len*).
Y	double byte character set (DBCS)	$HEX(2x*len*).
Z	mixed DBCS and single byte	$HEX(2x*len*).
2	halfword binary (aligned), unsigned	5.*dec*
2	halfword binary (aligned), signed	6.*dec*
4	fullword binary (aligned), unsigned	9.*dec*
4	fullword binary (aligned), signed	10.*dec*
8	doubleword binary (aligned), unsigned	17.0

* *len* is the value of the LENGTH attribute of the CA-DATACOM/DB field. *dec* is the value of the DECIMALS attribute of the CA-DATACOM/DB field.

Note that CA-DATACOM/DB numeric fields are copied into SAS character columns with a $HEX. format if they are too long to fit in a SAS numeric column. For example, a CA-DATACOM/DB field of type B with a length of 30 is copied into a SAS column with $HEX60. format. A field of type B with a length of 5 and *dec* of 2 is copied into a SAS column with 11.2 format. An error message appears if any precision is lost.

The maximum SAS format width is 200, so the SAS software uses 200 for CA-DATACOM/DB fields whose length exceeds 200.

You may want to change the default format whenever it does not seem appropriate for the values stored in the table. For example, a packed decimal field of length 4 and 2 decimal places would have a default SAS format of 7.2. A very large negative number with a decimal (such as -99,999.99) might not fit.

If SAS software discovers the output format is too small, it issues the following warning message to the error log: AT LEAST ONE W.D FORMAT WAS TOO SMALL FOR THE NUMBER TO BE PRINTED. THE DECIMAL POINT MAY BE SHIFTED BY THE BEST FORMAT. The message can occur, for example, when you invoke the PRINT procedure. If this message appears, you should specify a larger width.

The format determines how values in the SAS column are displayed; it does not affect how those values are stored. Their storage is determined by their CA-DATACOM/DB type and length. Therefore, you cannot replace a character format with a numeric format or vice versa.

If numeric values in the table are a lot smaller than their length implies, space on the output print line can be conserved by specifying smaller *w.* or *w.d* formats.

Each key is converted to one SAS character column, even if the key is numeric or has more than one component field. The length of a key becomes its default format width. You cannot change the format for a key.

If you assign a date format to a numeric field, be sure that you also specify the SAS date format in the DB Content field to indicate you are storing dates in your table.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS Interface to CA-DATACOM/DB Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 170.

SAS/ACCESS Interface to CA-DATACOM/DB Software: Reference, Version 8

ISBN 1–58025–545–0

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

1st printing, October 1999