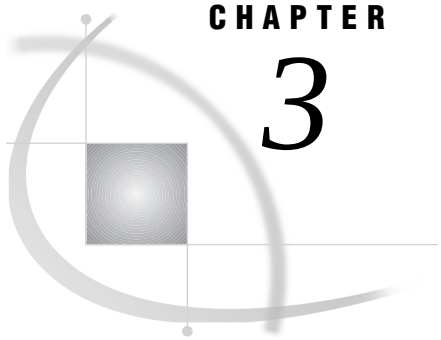

CHAPTER 3

Defining SAS/ACCESS Descriptor Files

Introduction

To use the SAS/ACCESS interface to CA-DATACOM/DB, you must define special files that describe CA-DATACOM/DB tables and data to the SAS System. These files are called SAS/ACCESS descriptor files. This chapter is a tutorial and uses examples to illustrate creating and editing these files, as well as using the ACCESS procedure to extract CA-DATACOM/DB data and place them in a SAS data file. (For complete reference information on the ACCESS procedure, see Chapter 6, "ACCESS Procedure Reference," on page 63.)

The examples in this chapter are based on the CA-DATACOM/DB table named CUSTOMERS. (See Appendix 3, "Data and Descriptors for the Examples," on page 125 to review the data in this table.) You will create an access descriptor file named MYLIB.CUSTS for that table. Then, you will create two view descriptor files, VLIB.USACUST and VLIB.CUSTADD, based on the access descriptor. The next sections provide some background on access and view descriptor files.

Understanding SAS/ACCESS Descriptor Files

The SAS System interacts with CA-DATACOM/DB through an interface view engine that uses SAS/ACCESS descriptor files created with the ACCESS procedure. There are two types of descriptor files:

- access descriptor files (member type ACCESS)
- view descriptor files (member type VIEW).

An access descriptor contains information about the CA-DATACOM/DB table you want to use. The information includes the table name, the field names, and their data types. You use the access descriptor to create view descriptors. Think of an access descriptor as being a master descriptor file for a single CA-DATACOM/DB table, because it usually contains a complete description of that table.

A view descriptor defines a subset of the data described by an access descriptor. You choose this subset by selecting particular fields in the CA-DATACOM/DB table, and you can specify selection criteria that the data must meet. For example, you may want to select two fields, LAST-NAME and CITY-STATE, and specify that the value stored in field CITY-STATE must be **AUSTIN TX**. You can also specify a sequence order for the data. After you create your view descriptor, you can use it in a SAS program to read data directly from the CA-DATACOM/DB table or to extract the data and place them in a SAS data file. Typically, for each access descriptor that you define, you have several view descriptors, each selecting different subsets of data.

Creating SAS/ACCESS Descriptor Files

The examples in this section illustrate creating a permanent access descriptor named MYLIB.CUSTS and two view descriptors named VLIB.USACUSTS and VLIB.CUSTADD. Begin by using the SAS LIBNAME statement to associate librefs with the SAS data libraries in which you want to store the descriptors. (See the SAS documentation for your operating system for more details on the LIBNAME statement.)

You can have one library for access descriptors and a separate library for view descriptors, or you can put both access descriptors and view descriptors in the same library. Having separate libraries for access and view descriptors helps you maintain data security by enabling you to separately control who can read and update each type of descriptor.

In this book, the libref MYLIB is used for access descriptors and the libref VLIB is used for view descriptors.

The ACCESS Procedure

You define descriptor files with the ACCESS procedure. You can define access descriptor files and view descriptor files in the same procedure execution or in separate executions. Within an execution, you can define multiple descriptors of the same or different types.

The following section shows how to define an access descriptor and multiple view descriptors in a single procedure execution. Examples of how to create the same descriptor files in separate PROC ACCESS executions are provided in Appendix 3, “Data and Descriptors for the Examples,” on page 125.

When you use a separate PROC ACCESS execution to create a view descriptor, note that you must use the ACCDESC= option to specify an existing access descriptor from which the view descriptor will be created.

Creating Access Descriptors and View Descriptors in One PROC Step

Perhaps the most common way to use the ACCESS procedure statements is to create an access descriptor and one or more view descriptors based on this access descriptor in a single PROC ACCESS execution. The following example shows how to do this. First an access descriptor is created (MYLIB.CUSTS). Then two view descriptors are created (VLIB.USACUST and VLIB.CUSTADD). Each statement is then explained in the order that it appears in the example program.

```
proc access dbms=datacom;
 create mylib.custs.access;
  user=demo;
  table=customers;
```

```
  assign = yes;
  drop contact;
  list all;
  extend all;
  rename customer = custnum telephone = phone
         streetaddress = street;
  format firstorderdate = date7.;
  informat firstorderdate = date7.;
  content firstorderdate = yymmdd6.;
  list all;

 create vlib.usacust.view;
  select customer state zipcode name
         firstorderdate;
  list view;
  extend view;

  subset where customer eq 1#;
  subset sort firstorderdate;
  list view;

 create vlib.custadd.view;
  select state zipcode country name city;
  list view;

  list all;

run;
```

proc access dbms=datacom;
invokes the ACCESS procedure for the SAS/ACCESS interface to CA-DATACOM/DB.

create mylib.custs.access;
identifies the access descriptor, MYLIB.CUSTS, that you want to create. The MYLIB libref must be associated with a SAS data library before you can specify this statement.

user=demo;
specifies a required CA-DATADICTIONARY userid. In this case, the user name is DEMO for the CA-DATACOM/DB table CUSTOMERS. The name is the 32–character entity-occurrence name of a PERSON entity in CA-DATADICTIONARY. The value entered is saved in the access descriptor and any view descriptor created from it. The user name and optional password (not used here) must have CA-DATADICTIONARY retrieval authority on six entity-types: DATABASE, FILE, RECORD, ELEMENT, KEY, and FIELD.

table=customers;
indicates the name of the CA-DATACOM/DB table that you want to use. The table name is required. The table name is a 32–character field that names an entity-occurrence of type RECORD in CA-DATADICTIONARY. (For CA-DATACOM/DB R8, the type is TABLE.) The combination of values in the TABLE statement and optional DATABASE and STATUS statements (not used here) must be unique.

assign = yes;
generates unique SAS column names based on the first eight non-blank characters of the CA-DATACOM/DB field names. The column names and attributes can be

changed in this access descriptor but not in any view descriptors created from this access descriptor.

Note that although the ASSIGN statement assigns names to the columns, it does not select them for inclusion in any view descriptors created from this access descriptor. You must select the fields in the view descriptor with the SELECT statement. Unless fields are dropped, they are automatically included in the access descriptor.

drop contact;

marks the CA-DATACOM/DB field with the name CONTACT as non-display. The CONTACT field is a simple field; therefore, it is the only DBMS column that is dropped. When the DROP statement indicates a compound field, which can consist of multiple simple and compound fields, all DBMS columns associated with the compound field are marked as non-display, unless otherwise specified with the OCCURS statement. Compound fields are identified by the word *GROUP* in their description in the LIST statement output.

Columns that are dropped also do not appear in any view descriptors created from this access descriptor.

list all;

lists the access descriptor's item identifier numbers, the CA-DATACOM/DB field names, the CA-DATACOM/DB level numbers, the SAS column names, and the SAS formats. You can use the item identifer as a field identifier in statements that require you to use the DBMS column name. The list is written to the SAS log. Any columns that have been dropped from display (using the DROP statement) have *NON-DISPLAY* next to them.

extend all;

lists information about the SAS columns in the access descriptor, including the informat, the DB content, and the number of times a field repeats. The list is written to the SAS log. When you are creating multiple descriptors, you can use the EXTEND statement before the next CREATE statement to list all the information about the descriptor you are creating.

rename customer = custnum telephone = phone streetaddress = street;

renames the default SAS column names associated with the CUSTOMER, TELEPHONE, and STREETADDDRESS fields to CUSTNUM, PHONE, and STREET, respectively. Specify the CA-DATACOM/DB field name or its positional equivalent from the LIST statement on the left side of the equal sign (=) and the new SAS name on the right. Because the ASSIGN=YES statement is specified, any view descriptors created from this access descriptor will automatically use the new names.

format firstorderdate = date7.;

changes the FIRSTORD SAS column from its default format to a new SAS format. The format specifies the way a value will be printed, in this case, as a date format. Specify the CA-DATACOM/DB field name or its positional equivalent from the LIST statement on the left side of the equal sign (=) and the new SAS format on the right. Because the ASSIGN=YES statement is specified, any view descriptors created from this access descriptor will automatically use the new format for the FIRSTORD column.

informat firstorderdate = date7.;

changes the FIRSTORD SAS column from its default informat to a new SAS informat. The informat specifies the way a value will be read, in this case, as a date informat. Specify the CA-DATACOM/DB field name or its positional equivalent from the LIST statement on the left side of the equal sign (=) and the new informat on the right. Because the ASSIGN=YES statement is specified, any

view descriptors created from this access descriptor will automatically use the new informat for the FIRSTORD column.

content firstorderdate = yymmdd6.;
specifies the SAS date format to use for the FIRSTORD SAS column. This format indicates the way date values are represented internally in the CA-DATACOM/DB table, in this case, **yymmdd**. Specify the CA-DATACOM/DB field name or its positional equivalent from the LIST statement on the left side of the equal sign (=) and the date format on the right. Because the ASSIGN=YES statement is specified, any view descriptors created from this access descriptor will automatically use this date format for the FIRSTORD column.

list all;
lists the item identifiers, the CA-DATACOM/DB field names, the SAS column names, and other SAS information in the access descriptor so you can see the modifications before proceeding with the next CREATE statement.

create vlib.usacust.view;
writes the access descriptor to the library associated with MYLIB and identifies the view descriptor, VLIB.USACUST, that you want to create. The VLIB libref must be associated with a libref before you can specify this statement.

select customer state zipcode name firstorderdate;
selects the CUSTOMER, STATE, ZIPCODE, NAME, and FIRSTORDERDATE fields for inclusion in the view descriptor. A SELECT statement is required to create the view, unless a RENAME, FORMAT, INFORMAT, or CONTENT statement is used.

list view;
lists the item identifiers, the DBMS column names, the SAS column names, and other SAS information associated with the CA-DATACOM/DB fields selected for the view. The list is written to the SAS log.

extend view;
lists detail information about the SAS columns in the view, including the informat, the DB content, and the number of times a field repeats. The list is written to the SAS log.

subset where customer eq 1#;
specifies you want to include only records with 1 as the first character in the CUSTOMER DBMS column.

subset sort firstorderdate;
specifies you want to order the records by the value of the FIRSTORDERDATE DBMS column.

list view;
lists the item identifiers, the DBMS column names, the SAS column names, and other SAS information associated with the view, to show the modifications.

create vlib.custadd.view;
writes view descriptor VLIB.USACUST to the library associated with VLIB and identifies a second view descriptor, VLIB.CUSTADD, that you want to create.

select state zipcode country name city;
selects the STATE, ZIPCODE, COUNTRY, NAME, and CITY fields for inclusion in the view descriptor.

list view;
lists the item identifiers, the DBMS column names, the SAS column names, and other SAS information associated with the CA-DATACOM/DB fields selected for the view.

list all;
lists updated SAS information for the fields in the access descriptor. Fields that were dropped have *NON-DISPLAY* next to the SAS column description. Fields selected in the VLIB.CUSTADD view descriptor have *SELECTED* next to them. Fields selected in VLIB.USACUST will not show as selected in the access descriptor. Selection information, including status and any selection criteria, are reset in the access descriptor for each new view descriptor. The list is written to the SAS log.

run;
writes the view descriptor when the RUN statement is processed.

Updating Descriptor Files

This section describes how to update existing descriptor files. You update access descriptor and view descriptor files with the UPDATE statement. You can edit the user and field information in the descriptor files.

When you update an access descriptor, the view descriptors based on this access descriptor are not updated automatically. You must re-create or update any view descriptors that you want to reflect the changes made to the access descriptor. That is, for some updates (such as dropping a field), the view descriptors are still valid, but they do not reflect the changes made in the access descriptor. In other situations (for example, if you edited the access descriptor to use a different userid or to add a password), the view descriptors would no longer be valid. A valid descriptor file can also be made useless by an update. For example, if an update to an access descriptor drops two of the four fields defined in a view descriptor, you may want to update or delete the view descriptor.

The following example updates access descriptor MYLIB.CUSTS to drop additional fields. The VLIB.USACUSTS and VLIB.CUSTADD view descriptors remain valid, however, you may want to update them to select new fields to replace those dropped as a result of the update.

```
proc access dbms=datacom;
 update mylib.custs.access;
  drop 3 7;
  list all;
run;
```

The statements are described below.

proc access dbms=datacom;
invokes the ACCESS procedure for the SAS/ACCESS interface to CA-DATACOM/DB.

update mylib.custs.access;
identifies the access descriptor, MYLIB.CUSTS, that you want to update. The MYLIB libref must be associated with a SAS data library before you can specify this statement.

drop 3 7;
marks the CA-DATACOM/DB fields associated with position 3 (STATEZIP) and position 7 (TELEPHONE) as non-display. STATEZIP is a compound (*GROUP*) field consisting of STATE and ZIPCODE. Dropping a group effectively drops the members of the group, so the STATE and ZIPCODE fields (which are selected in VLIB.USACUST and VLIB.CUSTADD) are marked as non-display as well.

list all;
lists updated SAS information for the fields in the access descriptor. Fields that were dropped have *NON-DISPLAY* next to the SAS column description. The list is written to the SAS log.

run;
writes the updated access descriptor when the RUN statement is processed.

Altering a CA-DATACOM table that has descriptor files defined can also cause these files to be out of date or invalid. If you add a field to a table, an access descriptor is still valid. However, if you delete a field or change its characteristics and that field is used in a view descriptor, the view will fail when executed. For more information, see "How Changing the CA-DATADICTIONARY Database Affects Descriptor Files" on page 110.

Using the ACCESS Procedure to Extract CA-DATACOM/DB Data

Although you can access CA-DATACOM/DB data directly in your SAS programs, it is sometimes better to extract the CA-DATACOM/DB data and place them in a SAS data file. For example, if you are using the same CA-DATACOM/DB data in several SAS jobs, it may be less resource-intensive to access extracted data in a SAS data file than to access a CA-DATACOM/DB table repeatedly. (See "Performance Considerations" on page 43 for other circumstances in which extracting data is the more efficient method.)

You can extract CA-DATACOM/DB data by using PROC ACCESS statement options. You can also extract data using the DATA step. (See Chapter 4, "Using CA-DATACOM/DB Data in SAS Programs," on page 23 for examples using the SQL procedure to extract CA-DATACOM/DB data and place them in a SAS data file.) Note that if you store view descriptors and SAS data files in the same SAS data library, you must give them unique member names.

Extracting with the PROC ACCESS Statement Options

To extract data using the PROC ACCESS statement options, submit the following SAS code:

```
proc access viewdesc=vlib.usacust out=mydata.usaout;
run;
```

VLIB.USACUST is the two-level name that specifies the libref and member name for the view descriptor you want to use for extracting data, in this case, USACUST. Note that VLIB.USACUST must already exist. MYDATA.USAOUT is the two-level name specifying the libref and member name for the output SAS data file.

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