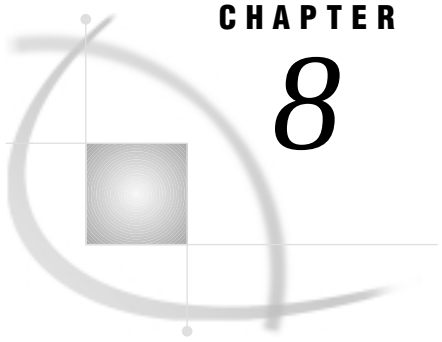

CHAPTER

8

DBLOAD Procedure Reference

Introduction **97**
DBLOAD Procedure Syntax **97**
Description **98**
PROC DBLOAD Statement Options **98**
Procedure Statements **99**
Creating Your Own View Descriptor **108**
Default SYSTEM 2000 Item Types **109**
Allocating the Database Files **109**
Adding Records That Lie on a Different Path **110**
Loading One SYSTEM 2000 Database from Another **110**

Introduction

This chapter provides reference information for the DBLOAD procedure. The DBLOAD procedure enables you to create and load a SYSTEM 2000 database using data from a SAS data file, from a view created with the SQL procedure, or from a SYSTEM 2000 database or another DBMS (using a view descriptor created with the ACCESS procedure).

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

DBLOAD Procedure Syntax

PROC DBLOAD <*options*>;
CREATE;
DBN= *database-name*;
ACCDESC= *libref.access-descriptor*;
DELETE *variable-identifier*
<*...variable-identifier-n*>;
INDEX *variable-identifier* = Y | N
<*...variable-identifier-n*= Y | N>;
LABEL;

LEVEL *variable-identifier = n*
<...*variable-identifier-n= n*>;

LIST *list-selection*;

LOAD;

QUIT;

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

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

S2KLEN *variable-identifier = n*
<...*variable-identifier-n = n*>;

S2KLOAD;

S2KMODE= M | S;

S2KPW= *password*;

VIEWDESC *libref.view-descriptor*;

WHERE *SAS-where-expression*;

Description

The DBLOAD procedure enables you to

- create a new database definition only
- create a new database definition and load data
- load new logical entries into an existing database
- insert new data records into existing logical entries.

The procedure constructs SYSTEM 2000 statements to create a new database definition. You can specify an optimized load process to load new logical entries into the database or an insert process to add new records into existing entries.

The DBLOAD procedure associates each SAS variable with a SYSTEM 2000 item and assigns a default name, item type, and picture to each item. You can change the component names as necessary. Also, by default, each item is non-key at level 0; however, you can change the item to be a key item, and you can specify a level number, which causes the procedure to create records below level 0. When you are finished customizing the items, the procedure creates the new database definition and loads the data unless you have specified that you do not want to load any data at this time.

When you load data into an existing database, you must specify an existing view descriptor. You can specify optimized loading for new logical entries; insert mode must be used for adding new records to existing logical entries.

The DBLOAD procedure can run in interactive line or batch mode. For efficiency considerations, you may want to use batch mode for loads that process large amounts of data.

PROC DBLOAD Statement Options

The following options can be used with the PROC DBLOAD statement:

DBMS= database-management-system
specifies the database management system you want to access. If you have only the Version 7 SAS/ACCESS interface to SYSTEM 2000 software installed on your machine, this option defaults to S2K. If you have more than one Version 7 SAS/

ACCESS interface installed, you must specify DBMS=S2K to access the SYSTEM 2000 data management system.

DATA= libref.SAS-data-set
specifies the input data set. A SAS data set can be either a SAS data file or a SAS data view. If the file is permanent, you must use its two-level name, *libref.SAS-data-set*. If you do not specify the DATA= option, the default is the last SAS data set created.

Procedure Statements

The statements you use with the DBLOAD procedure depend on whether you are creating a new database in which to load data or whether you are appending data to an existing database, and even then, most are optional.

The following statements are required:

- DBN and S2KPW when creating and loading a new database
- VIEWDESC when appending data to an existing database
- LOAD for both loading and appending
- CREATE for creating a database without loading any data.

Of the remaining statements, most are allowed only when creating a new database; they produce warnings when used with an existing database. If the view descriptor exists, PROC DBLOAD assumes you are trying to add data to an existing database; therefore, it will not accept the statements used when creating a database.

The statements listed in Table 8.1 on page 99 apply only to creating a database.

Table 8.1 DBLOAD Statements Used When Creating a Database

ACCDESC	INDEX	RESET
CREATE	LABEL	S2KMODE
DBN	LEVEL	S2KPW
DELETE	RENAME	S2KLEN

ACCDESC

Assigns a name to the access descriptor for a new database

Optional statement

Applies to: New database

Syntax

ACCDESC|AD|ACCESS= *libref.access-descriptor*;

Details The ACCDESC= statement specifies an access descriptor name for the new database. If the member name already exists, the DBLOAD procedure will not create the new database.

The DBLOAD procedure always creates an access descriptor file when it creates a database. WORK.<*database*>.ACCESS is the default name, where *database* contains the first seven characters of the new database name. It also creates a view descriptor that matches the access descriptor. (See also "VIEWDESC" on page 107.)

Note: When you are creating a new database and you want to issue the ACCDESC= statement, it must not be the first statement in the procedure. It must follow some statement that is allowed only for creating a new database so that the DBLOAD procedure knows your intent. △

AD= and ACCESS= are equivalent to the ACCDESC= statement.

CREATE

Creates a database definition

Optional statement

Applies to: New database

Syntax

CREATE;

Details The CREATE statement indicates you want to create a SYSTEM 2000 database definition, but not load any data. By default, the DBLOAD procedure expects you to load data.

DBN

Specifies the database to be created

Required statement

Applies to: New database

Syntax

DBN|DB= *database-name*;

Details The DBN statement specifies the name of the database to be created. The database name must be a valid SYSTEM 2000 database name, from one to sixteen characters long. Database names longer than 16 characters will be truncated with no error message. If the database name contains embedded blanks or special characters, enclose it in single quotes. The slash (/), colon (:), and equal (=) special characters are not allowed.

This statement is required when you are creating a new database. A database with the same name must not already exist.

The software uses the first seven characters of the database name as part of the DDname for the database files. Any restrictions imposed by the operating system on DDnames also apply to the characters used in the database name.

For single-user jobs, you must allocate your files to your SAS session. For the Multi-User environment, the database files can be allocated when the Multi-User software is initialized or (Release 12.0 and later) the files can be dynamically allocated during execution using the ALLOC command.

DB= is an alias for the DBN= statement.

DELETE

Does not load specified variables into the new database

Optional statement

Applies to: New database

Syntax

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

Details The DELETE statement indicates you want to delete (drop) the specified variables from the load. By default, all SAS variables will be loaded unless you specify a DELETE statement.

The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file. For example, if you want to delete the third variable, issue the following statement:

```
delete 3;
```

You can delete as many variables as you want in one DELETE statement. If you delete more than one variable, separate the identifiers with spaces, not commas.

Note: Even if you delete a variable from the table, this does not change the positional equivalents of the variables. For example, if you delete the second variable, the third variable is still referenced by the number 3, not 2. △

INDEX

Optional statement

Applies to: New database

Syntax

INDEX *variable-identifier* = Y | N

<...*variable-identifier-n*= Y | N>;

Details The INDEX statement indicates the key or non-key status of items in the SYSTEM 2000 database. Y means the item will be indexed (key); N means the item will not be indexed (non-key). The default is non-key (N).

The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file.

LABEL

Causes DBMS column names to default to SAS labels

Optional statement

Applies to: New database

Syntax

LABEL;

Details The LABEL statement specifies that you want the SYSTEM 2000 item names to default to the 40-character SAS variable labels. If a variable has no label, the 8-character SAS variable name is used.

LEVEL

Optional statement

Applies to: New database

Syntax

LEVEL *variable-identifier = n*
<...*variable-identifier-n=n*>;

Details The LEVEL statement allows you to specify a level number for one or more variables that will become items in the SYSTEM 2000 database. The default is level 0. If you specify any items below level 0, the DBLOAD procedure automatically defines the appropriate schema records. The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file. n is an integer from 0 through 9.

LIST

Lists information about the variables to be loaded

Optional statement

Applies to: New and existing database

Syntax

LIST *list-selection*;

Details The LIST statement causes a list of information to be displayed for all input variables, along with the current options, such as key or non-key and level number. The default destination of the list is the SAS log. The list-selection can be one or more of the following:

ALL	causes all information for the load to be listed.
FIELDS | ITEMS	causes all SYSTEM 2000 items for the load to be listed. ITEMS is equivalent to FIELDS.
variable-identifier	causes only one line to be listed, with the information about the specified variable.

The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file. For example, if you want to list the information for the item associated with the third SAS variable, issue the following statement:

```
list 3;
```

You can use one or more of these options in the LIST statement in any order. Here is an example:

```
list 3 fields 4;
```

This statement lists the information for the third SAS variable, followed by all the items in the data file, followed by the information for the fourth SAS variable.

LOAD

Executes the load operation

Optional statement

Applies to: New and existing database

Syntax

LOAD;

Details The LOAD statement specifies that you want to execute the DBLOAD procedure.

QUIT

Terminates the DBLOAD procedure

Optional statement

Applies to: New and existing database

Syntax

QUIT|END|EXIT;

Details The QUIT statement specifies you want to exit the procedure without further processing. END and EXIT are aliases for the QUIT statement.

RENAME

Renames DBMS columns

Optional statement

Applies to: New database

Syntax

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

Details The RENAME statement specifies you want to change the names of the SYSTEM 2000 items associated with the listed SAS variables. The new component names go into the access descriptor and the view descriptor that are created for the new database.

The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file. For example, if you want to rename the item associated with the third SAS variable, issue the following statement:

```
rename 3='employee name';
```

The name must be a valid SYSTEM 2000 component name. If the item name includes embedded blanks or invalid SAS name characters, such as the pound sign (#) or hyphen (-), you must enclose the item name in single quotes.

The RENAME statement enables you to include variables you have previously deleted. For example, if you first issued the statement DELETE 3, then issuing

RENAME 3=XYZ includes the third variable and assigns it the name XYZ and the default item type.

If you do not use the RENAME statement, all SYSTEM 2000 item names default to the corresponding SAS names or to the SAS labels if you specified the LABEL statement. You can list as many variables as you want in one RENAME statement. The RENAME statement overrides the LABEL statement for the items that are renamed.

RESET

Resets column names and data types to their default values

Optional statement

Applies to: New database

Syntax

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

Details The RESET statement resets the items associated with the listed SAS variables to their defaults. If you use ALL, all items are reset to the defaults and deleted items will be restored with default values. Item names default to SAS variable names (or labels), item types are generated from the SAS variable formats, and all items are non-key at level 0.

The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file. For example, if you want to reset the item associated with the third SAS variable, issue the following statement:

```
reset 3;
```

You can reset as many items as you want in one RESET statement. ALL means all previous RENAME, DELETE, INDEX, LEVEL, and S2KLEN statements will be ignored.

S2KLEN

Changes the SAS variable length of DBMS column names

Optional statement

Applies to: New database

Syntax

S2KLEN *variable-identifier = n*
<*...variable-identifier-n = n*>;

Details The S2KLEN statement allows you to change the SYSTEM 2000 picture for a CHARACTER or TEXT type item. *n* is an integer value from 1 to 250. The value of n is used in the definition of the new database, for example, CHAR X(10). If you do not specify the length of a CHARACTER or TEXT item, the SAS variable length is used.

The variable-identifier can be either the SAS variable name or the positional equivalent in the LIST output, which is the number that represents the variable's place in the data file.

The main reason for changing the picture is to allow overflow when the SAS length is greater than 4. A SYSTEM 2000 picture equal to or greater than 4 allows overflow of CHARACTER or TEXT type data values. For example, if the length of a SAS variable is 80 and you set the SYSTEM 2000 picture to 4, the entire value goes into overflow.

S2KLOAD

Turns on optimized load mode processing

Optional statement

Applies to: New and existing database

Syntax

S2KLOAD;

Details The S2KLOAD statement controls whether SYSTEM 2000 software uses optimized load mode processing. You can use the optimized load for the initial load or for incremental loads that involve adding entirely new logical entries. However, when you are inserting new records into existing entries, you cannot use optimized loading; the records are inserted below existing records. The default is insert mode.

S2KMODE=

Optional statement

Applies to: New database

Syntax

S2KMODE|S2KMD= M|S;

Details The S2KMODE= statement specifies your mode of accessing SYSTEM 2000 software. M means the database files are in the SYSTEM 2000 Multi-User environment. S means the database is in a single-user environment, that is, a database in your SAS program environment. S is the default.

S2KMD= is an alias for the S2KMODE= statement.

S2KMODE= is also a data set option for input data views for SAS procedures. However, you cannot use it as a data set option for the DBLOAD procedure. For more details about the S2KMODE= data set option, see "Data Set Options" on page 128.

S2KPW=

Assigns a database password

Required statement

Applies to: New database

Syntax

S2KPW= *password*;

Details The S2KPW= statement specifies the master password for the database that will be created.

The password must be acceptable to SYSTEM 2000 software. The password can be one to four characters long, with no embedded blanks, and optionally enclosed in single quotes. Passwords longer than four characters will be truncated with a warning message. If you specify a special character for the password, it must be a single character (that is, a one-character password) enclosed in single quotes.

S2KPW= is also a data set option for input data views for SAS procedures. However, you cannot use it as a data set option for the DBLOAD procedure. For more details about the S2KPW= data set option, see "Data Set Options" on page 128.

VIEWDESC

Assigns a name to the view descriptor for a new database

Required statement for an existing database

Optional statement for a new database

Syntax

VIEWDESC|VIEW|VD *libref.view-descriptor*;

Details The VIEWDESC statement identifies the view descriptor for the SYSTEM 2000 database to be created or loaded.

For an existing database, the VIEWDESC= statement is required since it contains the database name and identifies the password, along with the components in the view.

When you are creating a new database, the DBLOAD procedure creates a view descriptor. The default name is WORK.<*database*>.VIEW, where *database* contains the first seven characters of the new database name. The view descriptor will match the

access descriptor. Use the VIEWDESC= statement to specify the libref and member name for a permanent view descriptor. If the member name for the view descriptor already exists, the DBLOAD procedure will not create a new database. (See also "ACCDESC" on page 99.)

Note: When you are creating a new database and you want to issue the VIEWDESC= statement, it must not be the first statement in the procedure. It must follow some statement that is allowed only for creating a new database so that the DBLOAD procedure knows your intent. △

VIEW= and VD= are aliases for the VIEWDESC= statement.

WHERE

Optional statement

Applies to: New and existing database

Syntax

WHERE *SAS-where-expression*;

Details The WHERE statement indicates you want to subset your input data. The SAS-where-expression must be a valid SAS WHERE statement. Here is a simple example of a WHERE statement.

```
where custname='JONES, APRIL';
```

This statement loads only those observations with JONES, APRIL in the SAS variable CUSTNAME.

Use the SAS variable names in the WHERE statement, not the SYSTEM 2000 component names. For more details on the syntax of the SAS WHERE statement, see *SAS Language Reference: Dictionary*.

Creating Your Own View Descriptor

When the DBLOAD procedure creates a new database, it always creates an access descriptor and a view descriptor that matches the access descriptor. Their default names are WORK.database-name.ACCESS and WORK.database-name.VIEW, respectively.

If you do not like the default descriptors that the DBLOAD procedure creates, issue the CREATE statement and invoke the ACCESS procedure to create your own specific view descriptor; then return to the DBLOAD procedure, specify your view descriptor, and load your data.

You must ensure that incoming SAS variables match the SYSTEM 2000 items in your view descriptor. You can have more components in your view descriptor than in the SAS data file or vice versa. The procedure matches input variables with the variables in the view descriptor by SAS names. If your SAS names do not match, do one of the following:

- Use the ACCESS procedure to create a view descriptor that matches the SAS data file.

- Use the RENAME data set option on the DATA= argument. Use the KEEP or DROP option in the DATA= argument to limit the SAS variables that are inspected.

Default SYSTEM 2000 Item Types

Here are the default conversions of SAS formats to SYSTEM 2000 item types. You cannot change them. However, you can alter the formats of the input SAS variables with the DATASETS procedure and its MODIFY and FORMAT statements if you want to affect the behavior of the type conversions. These formats will be saved in the access descriptor and view descriptor.

Table 8.2 Default SYSTEM 2000 Item Types and Pictures

SAS Format	SYSTEM 2000 Type
$*w*.	CHAR X(*n*)
$CHAR*w*.	TEXT X(*n*)
any date format	DATE
w.	INTEGER 9(*n*)
w.d	DECIMAL 9(*n*-*d*-1).9(*d*)
DOLLAR*w.d*	MONEY 9(*n*-*d*-1).9(*d*)
E*w*.	
if *n*. < 8	REAL
if *n*. >= 8	DOUBLE
$HEX*w*.	UNDEFINED X(*n*)

Note: *n* is the length of the SAS variable. The value of *w* is ignored. △

If there is no SAS format, a character variable becomes a type CHARACTER item (with a picture equal to the variable length or the value you specified in an S2KLEN statement), numeric variables that are four bytes become type REAL, and numeric with eight bytes become type DOUBLE. The formats saved in the access descriptor and view descriptor will be $*w*. for character and BEST12. for numeric.

Allocating the Database Files

In a single-user job, you must allocate the appropriate database files in your SAS session prior to invoking the DBLOAD procedure. For a Multi-User environment, the database files must already exist and can be allocated when the Multi-User software is initialized or (Release 12.0 or later) they can be dynamically allocated during execution using the ALLOC command. If a database already exists, it will not be released; SYSTEM 2000 software returns a message and the DBLOAD procedure terminates.

Adding Records That Lie on a Different Path

The DBLOAD procedure allows you to have records at multiple levels, but they must be on the same path. If you have disjoint schema records, you must create the database definition outside of the DBLOAD procedure. Use the ACCESS procedure to create the access descriptor and view descriptors. You can then use the DBLOAD procedure to load data one path at a time in incremental loads.

Loading One SYSTEM 2000 Database from Another

To load one SYSTEM 2000 database from another, you use a view descriptor as input, but one restriction applies. Both databases cannot be in the same execution environment if you request optimized load processing. In this situation, one database must be in a single-user environment and the other must be in the Multi-User environment. (Optimized load puts the database under exclusive use, which excludes access to other databases in that environment until exclusive use is relinquished.)

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

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

ISBN 1–58025–549–3

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

1st printing, October 1999