CHAPTER

3

Using SAS Files

Introduction to SAS Files

This section briefly reviews SAS files, taking into account that your SAS files are stored in Windows. For additional information about SAS files, see *SAS Language Reference: Dictionary*.

What Is a SAS File?

The SAS System creates and uses a variety of specially structured files called *SAS files*. Although Windows manages the file for the SAS System by storing it, the operating system cannot process it. For example, you can list SAS files with the Windows Explorer, but you cannot use the Windows Notepad to edit SAS files. SAS files are different from external files. While external files can be processed by SAS statements and commands, they are not managed by the SAS System.

SAS files usually reside in SAS data libraries. Under Windows, a SAS library is simply a *named collection of SAS files within one or more Windows folders that the SAS System can access*. Each SAS data library has an access engine associated with it the first time that a file in the library is accessed. The engine name specifies the access method that the SAS System uses to process the files in the data library. SAS data libraries are described in detail in *SAS Language Reference: Dictionary*.

Various engines enable the SAS System to access different formats or versions of SAS files and other vendors' files. For this reason, the SAS System is said to have Multiple Engine Architecture. Multiple Engine Architecture, combined with conversion utilities, provides access to Version 8 files and SAS files created with previous releases of the SAS System (back to Version 5), whether they were created under Windows or other operating systems. Multiple Engine Architecture also provides access to files created by other vendors' products, including database files.

The following sections highlight information you need in order to create and use SAS files with the various engines under Windows.

Types of SAS Files

SAS files are stored in SAS data libraries and are referred to as *members* of a library. Each member has a *member type*. The SAS System distinguishes between SAS files and external Windows files in a folder by using unique file extensions. The SAS System assigns certain file extensions to a general set of SAS member types. Table 3.1 on page 79 lists the Windows file extensions and their corresponding SAS member types for the V6, V7, and V8 engines. For more information about engines, see "Multiple Engine Architecture" on page 83.

Table 3.1 Windows File Extensions and Their Corresponding SAS Member Types

V6 File Extension	V7 and V8 File Exension (Short)	V7 and V8 File Extension (Long)	SAS Member Type	Description
.sas	.sas	.sas	none	SAS program
.ss2	.ss7	.sas7bpgm	Program	stored program (DATA step)
.lst	.lst	.lst	none	output file
.log	.log	.log	none	log file
none	.st7	.sas7baud	Audit	audit file
.sd2	.sd7	.sas7bdat	Data	data set
.sv2	.sv7	.sas7bvew	View	data set view
.si2	.si7	.sas7bndx	Index	data set index. Indexes are stored as separate files but are treated by the SAS System as integral parts of the SAS data file.
.sc2	.sc7	.sas7bcat	Catalog	SAS catalog
.sa2	.sa7	.sas7bacs	Access	access descriptor file
.sf2	.sf7	.sas7bfdb	FDB	consolidation database file
.sm2	.sm7	.sas7bmdb	MDDB	multi-dimensional database file
none	.s7m	.sas7bdmd	DMDB	data mining database file
none	.sr7	.sas7bitm	Itemstor	item store file
.su2	.su7	.sas7butl	Utility	utility file
.sp2	.sp7	.sas7bput	Utiltiy	permanent utility
.stx	.stx	none	none	transport file
none	.sb7	.sas7bbak	none	backup file

CAUTION:

Do not change the file extension of a SAS file; doing so can cause unpredictable results. The file extensions assigned by the SAS System to SAS files are an integral part of how the SAS System accesses these files. Also, you should not change the filename of a SAS file using operating system commands. If you want to change the name of a

SAS file, use the DATASETS procedure or select the file in the SAS Explorer window and select

Edit ▶ Rename...

△

Note: You may see files with other file extensions in your WORK and SASUSER data libraries. Most of these are temporary utility files that you do not need to access directly; be sure not to delete any of them during your SAS session.

If for some reason your SAS session ends abnormally, you might need to delete these files, outside of the SAS System, in order to regain disk space. △

Using Short or Long File Extensions in SAS Libraries

Version 8 libraries can either be short file extension libraries or long file extension libraries. Although the Windows operating environment and the SAS System for Version 8 support long file names, short file extension libraries are necessary for accessing libraries residing on servers that supports only short file extensions.

You can specify whether the library supports short or long file extensions on the LIBNAME statement. For example, if your SAS library is on a server mapped as the S drive and the server file system supports only short file extensions, your libname statement would look similar to this:

```
libname mylib 's:\sasv6' shortfileext;
```

For information on specifying short or long file extensions using the LIBNAME statement, see "LIBNAME" on page 384.

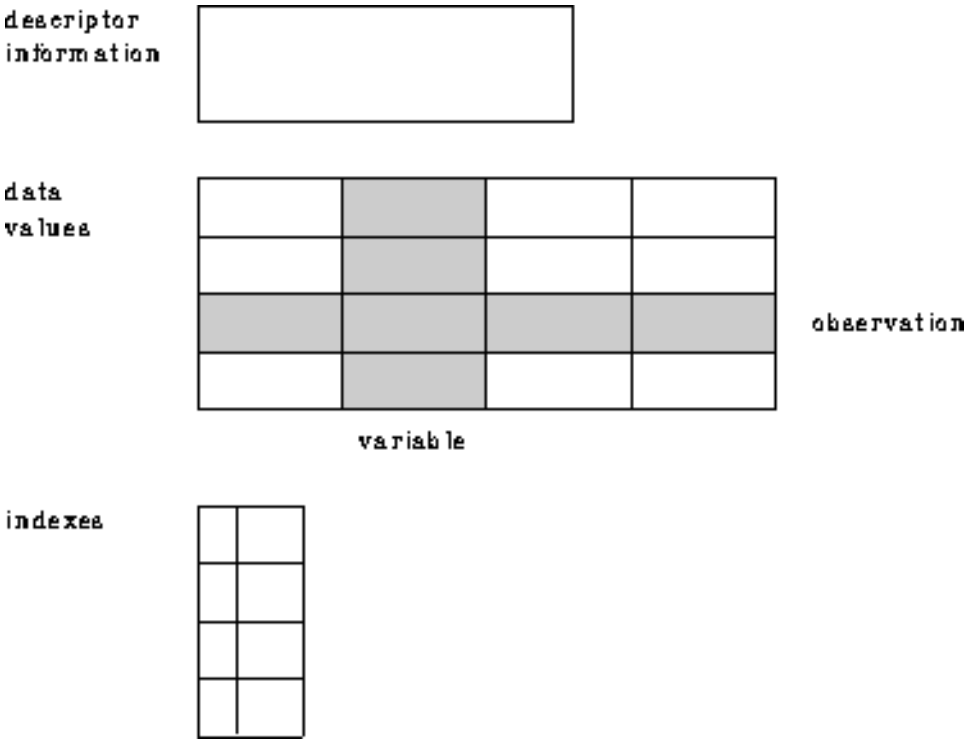
If SAS is not able to create a file with a long file extension the first time it writes to a library, then the library supports only files with short file extensions. If you specify a file with a long file extension for a library that supports only short file extensions, an error message informs you that the member name is too long for the system.

SAS Data Sets (Member Type: Data or View)

SAS data set is an umbrella term for SAS data files and SAS data views, which are both discussed here. This section provides a brief overview of the concept of SAS data sets. For complete details, see the data sets section in *SAS Language Reference: Concepts*.

Logically, a SAS data set consists of two types of information: descriptor information and data values. The descriptor information includes such things as data set name, data set type, data set label, and number of variables, as well as the names and labels of the variables in the data set, their types (character or numeric), their length, their position within a record, and their formats. The data values contain values for the variables. A SAS data set can be visualized as a table consisting of rows of observations and columns of variable values. Figure 3.1 on page 81 illustrates the SAS data set model.

Figure 3.1 SAS Data Set Model



SAS data files (member type: Data)

The *SAS data file* is probably the most frequently used type of SAS file. SAS data files have a SAS member type of Data and are created in the DATA step and by certain SAS procedures such as the RANK procedure in base SAS software.

The SAS System defines two types of SAS data files, native and interface. Native data files store data values and descriptor information, as described earlier, in files formatted by the SAS System. These are the SAS data sets you may be familiar with from previous versions of the SAS System under other operating systems. In the SAS System under Windows, native SAS data files can be indexed. The *index* is an auxiliary file created in addition to the SAS data file. The index provides fast access to records within a SAS data file through a variable or key. Indexes are stored as separate files but are treated by the SAS System as integral parts of the SAS data file.

The second type of data file is the interface SAS data file. These files store data in a file formatted by other software. Examples of interface SAS data files are BMDP, OSIRIS and SPSS files, which the SAS System can access as read-only files. For more information, see "Reading BMDP, OSIRIS and SPSS Files" on page 99.

In most cases, the maximum file size for a SAS data set is 2 gigabytes (GB). However, if you run the SAS System under Windows NT and store your data on a volume formatted with the Windows NT file system (NTFS), you can create and store data sets larger than 2GB. For more information about this feature and its uses, see "Using Large Data Sets with Windows NT and NTFS" on page 82.

For information about the size limitation of a data set under Windows, see "Length and Precision of Variables under Windows" on page 481.

SAS data views (member type: View)

SAS data views have a member type of View. They describe data values and tell the SAS System where to find the values, but they do not contain the actual data values themselves.

Views may be of two kinds, native or interface. A native SAS data view is created with the SQL procedure or with the DATA step and describes a subset or combination of the data in one or more SAS data files or SAS data views. For information on SQL views, see the *SAS Procedures Guide*. For information on DATA step views, see *SAS Language Reference: Dictionary*.

Interface SAS data views contain descriptor information for data formatted by other software products, for example, a database management system. Such a view is created with the ACCESS procedure in SAS/ACCESS software. For more information, see *SAS/ACCESS Software for PC File Formats: Reference* and other available SAS/ACCESS documentation.

SAS Catalogs (Member Type: Catalog)

A *SAS catalog* is a special type of SAS file that can contain multiple entries. You can keep different types of entries in the same SAS catalog. For example, catalogs can contain windowing applications, key definitions, toolbox definitions, SAS/GRAPH graphs, SAS/IML matrices, and so on.

If you want to use Version 8 to access catalogs created with earlier releases of the SAS System for Windows 95 or Windows NT, you must first convert the catalogs from the earlier releases to Version 8 format before you can use them in a Version 8 SAS program.

For more information on how to convert SAS catalogs, see *Moving and Accessing SAS Files across Operating Environments*.

SAS Stored Program Files (Member Type: Program)

A *stored program file* is a compiled DATA step generated by the Stored Program Facility. For more information about this type of SAS file, see *SAS Language Reference: Concepts*.

Access Descriptor Files (Member Type: Access)

Descriptor files created by the ACCESS procedure in SAS/ACCESS software have a member type of ACCESS and are used when creating interface SAS data views. Descriptor files describe the data formatted by other software products supported by the SAS System under Windows. For more information, see *SAS/ACCESS Software for PC File Formats: Reference* and other available SAS/ACCESS documentation.

Using Large Data Sets with Windows NT and NTFS

If you run SAS under Windows NT and store your data on a disk volume formatted with the Windows NT file system (NTFS), SAS automatically takes advantage of the 64-bit file I/O features. As a result, you can create, sort, and subset data sets greater than the 2 gigabyte size limit placed on other Windows environments. (The size limit for a SAS data set under Windows NT with NTFS is 4 giga-gigabytes, or 2^{62} bytes.)

Note that while you can access the full data set from SAS under Windows NT, other users running SAS under Windows 95 are able to access only the first 2 gigabytes (thus causing unpredictable results).

Multiple Engine Architecture

All permanent and temporary SAS files are stored in SAS data libraries. To use a SAS data library in your SAS session, you must assign a *libref* (library reference) and an engine to the data library. The libref is the name you use to refer to the data library during a SAS session or job. You can programmatically define it with an environment variable or with the LIBNAME statement or function. For a complete explanation of librefs, see *SAS Language Reference: Concepts*. The SAS Explorer window provides an easy way to manage all of your SAS files, including librefs. For information about working with SAS files in the SAS Explorer window, see *Getting Started with the SAS System*. For information on using librefs in the Windows environment, see "Using Data Libraries" on page 85

A *SAS data library* is a collection of SAS files within a Windows folder or a concatenation of folders. Although the folder can contain files that are not managed by the SAS System, only SAS files are considered part of the SAS data library. Any Windows folder can be treated as a SAS data library.

Access methods called engines provide access to many formats of data, giving the SAS System a *Multiple Engine Architecture*. Engines apply only to SAS data sets.

The engine identifies the set of routines that the SAS System uses to access the files in the data library. With this architecture, data can reside in different types of files, including SAS data files and data formatted by other software products, such as database management systems. By using the appropriate engine for the file type, the SAS System can write to or read from the file. For some types of files, you need to tell the SAS System what engine to use. For others, SAS automatically chooses the appropriate engine. For more details about engines and Multiple Engine Architecture, see *SAS Language Reference: Concepts*.

Engines are of two basic types, library and view. *Library engines* control access at the SAS data library level and can be specified in the LIBNAME statement or function. *View engines* enable the SAS System to read SAS data views described by the DATA step, SQL procedure, or SAS/ACCESS software. The use of SAS view engines is automatic because the name of the view engine is stored as part of the descriptor portion of the SAS data set. You cannot specify a view engine in the LIBNAME statement or function.

Library Engines

SAS has two types of library engines: native and interface. These engines support the SAS data library model. Library engines perform several important functions, including determining fundamental processing characteristics. For a more detailed description of library engines, see *SAS Language Reference: Dictionary*. For examples of using library engines, see "Using Data Libraries" on page 85.

Native Library Engines

Native library engines are those engines that access forms of a SAS file created and maintained by the SAS System. Native library engines include the default engine, the compatibility engine, and the transport engine. Table 3.2 on page 84 lists the acceptable names (and nicknames) for these engines.

Table 3.2 Native Library Engines

	Engine Names	Description
default	V8, BASE	accesses Version 8 data files
Version 7 compatibility	V7	accesses Version 7 data files
Release 6 compatibility	V6	accesses any data file created by Release 6.08 through Release 6.12
Release 6.03 and Release 6.04 compatibility	V604	accesses data files created by Release 6.03 and Release 6.04
transport	XPORT	accesses transport files

When using the default engine, choose which name, V8 or BASE, that you use in your SAS jobs with an eye to the future. If your application is intended for Version 8 of the SAS System only, and you do not want to convert it to later releases, use the name V8. If, however, you plan to convert your application to new releases of the SAS System, use the name BASE because that refers to the latest default engine. Using the name BASE makes your programs easy to convert. The engine name BASE does not refer to base SAS software; rather, it refers to the base, or primary, engine. The BASE engine can be used with more than the base SAS software product.

Note: This book uses the term *default engine* to refer to the V8 engine. The V8 engine is the default engine for accessing SAS files under Version 8 of the SAS System unless the default engine is changed with the ENGINE system option. To see the value of the ENGINE system option, select

Tools ► Options ► System

to open the SAS System Options window. Then select

Files ► SAS Files

The Engine system option displays the default engine for SAS data libraries. △

Interface Library Engines

Interface library engines support access to other vendors' files. These engines allow read-only access to BMDP, OSIRIS, and SPSS files. You must specify as part of the LIBNAME statement or function the name of the interface library engine that you want. Table 3.3 on page 85 lists the interface engine names:

Table 3.3 Interface Library Engines

Name	Description
BMDP	allows read-only access to BMDP files
OSIRIS	allows read-only access to OSIRIS files
SPSS	allows read-only access to SPSS files

For more information about these engines, see "Reading BMDP, OSIRIS and SPSS Files" on page 99.

Rules for Determining the Engine

If you do not specify an engine name in a LIBNAME statement or function, the SAS System attempts to determine the engine (either the default or a compatibility engine) that should be assigned to the specified data library libref. Under Windows, the SAS System looks at the file extensions that exist in the given folder and uses the following rules to determine which engine should be assigned to the libref:

- If the folder contains SAS data sets from only one of the supported native library engines (not including XPORT), the libref is assigned to that engine.
- If there are no SAS data sets in the given folder, the libref is assigned to the default engine.
- If the folder contains SAS data sets from more than one engine, this is called a mixed mode library. The libref is then assigned to the default engine. A message is printed in the SAS log informing you the libref is assigned to a mixed mode library.

Note: It is always more efficient to specify the engine name than to have the SAS System determine the correct engine. △

You can use the ENGINE system option to specify the default engine the SAS System uses when it detects a mixed mode library or a library with no SAS files. The valid values for the ENGINE option are V8, V7, V6, and BASE. By default, the ENGINE option is set to V8. For more information, see the system option"ENGINE" on page 424.

Using Data Libraries

The libref is a label or alias that is temporarily assigned to a folder so that the storage location (the full path, including drive and folder) is in a form that is recognized by the SAS System. A libref exists only during the session in which it is created. It is a logical concept describing a physical location, rather than something physically stored with the file. A libref follows the same rules of syntax as any SAS name. See the SAS language rules section in *SAS Language Reference: Dictionary* for more information on SAS naming conventions.

There are several ways to specify a libref:

- Use the New Library dialog box which is described in SAS System Help.
- Use the LIBNAME statement or function as described in "Assigning SAS Libraries Using the LIBNAME Statement or Function" on page 86
- Define an environment variable as described in "Assigning SAS Libraries Using Environment Variables" on page 88.

Note: You can eliminate the LIBNAME statement by directly specifying the drive name and the library name within quotes. An example follows:

```
data "d:\a";
```

△

Accessing the New Library Dialog Box Using the Graphical-User Interface

You can assign librefs using the graphical-user interface (GUI) either with the New Library toolbar icon (the file cabinet), the LIBASSIGN command, or from within Explorer. All of these options open the New Library dialog box where you can specify librefs and engines for multiple folders.

To open the New Library dialog box using the toolbar, click on the New Library icon which looks like a file cabinet.

To open the New Library dialog box using a command, type *libassign* in the command box.

To open the New Library dialog box using Explorer:

1. Select the Library folder.
2. Select **New...** from the **File** menu or right-mouse click the Library folder and select **New...** from the pop-up menu to open the New... dialog box.
3. Select the **Library** folder and click **OK**.

Note: When a second Explorer window is open on the right side of the SAS workspace, you can by-pass the New... dialog box to open the New Library dialog box if you right-mouse click on the **Libraries** folder and select **New...**. △

For more information on the New Library window and Explorer, see the SAS System Help.

Assigning SAS Libraries Using the LIBNAME Statement or Function

You can use the LIBNAME statement or function to assign librefs and engines to one or more folders, including the working folder. The examples in this section use the LIBNAME statement. For information on the LIBNAME function, see *SAS Language Reference: Dictionary*.

The LIBNAME statement has the following basic syntax:

LIBNAME *libref* <*engine-name*> '*SAS-data-library*'

An explanation of all the arguments in this statement can be found in "LIBNAME" on page 384 and *SAS Language Reference: Dictionary*.

You can also use the LIBNAME function within your SAS programs to assign librefs. For more information about the LIBNAME function, see *SAS Language Reference: Dictionary*.

CAUTION:
The words CON, NUL, LPT1 - LPT9, COM1 - COM9, and PRN are reserved words under Windows. Do not use these reserved words as librefs. △

Assigning a Libref to a Single Folder

If you have Version 8 SAS data sets stored in the C:\MYSASDIR folder, you can submit the following LIBNAME statement to assign the libref TEST to that folder:

```
libname test V8 'c:\mysasdir';
```

This statement indicates that Version 8 SAS files stored in the folder C:\MYSASDIR are to be accessed using the libref TEST. Remember that the engine specification is optional.

Assigning a Libref to the Working Folder

If you want to assign the libref MYCURR to your current SAS System working folder, use the following LIBNAME statement:

```
libname mycurr '.';
```

Assigning a Libref to Multiple Folders

If you have SAS files located in multiple folders, you can treat these folders as a single SAS data library by specifying a single libref and concatenating the folder locations, as in the following example:

```
libname income ('c:\revenue','d:\costs');
```

This statement indicates that the two folders, C:\REVENUE and D:\COSTS, are to be treated as a single SAS data library. When you concatenate SAS data libraries, the SAS System uses a protocol (a set of rules) for accessing the libraries, depending on whether you are accessing the libraries for read, write, or update.

Furthermore, you may concatenate multiple libraries by specifying only their librefs, as in the following example:

```
libname sales (income revenue);
```

This statement indicates that two libraries that are identified by librefs INCOME and REVENUE are treated as a single SAS data library whose libref is SALES.

For more information, see "Understanding How Concatenated SAS Data Libraries Are Accessed" on page 91 and *SAS Language Reference: Dictionary*.

Note: The concept of library concatenation also applies when specifying system options, such as the SASHELP and SASMSG options. For information on how to specify multiple folders in options such as these, see "Syntax for Concatenating Libraries in SAS System Options" on page 397. △

Assigning Engines

If you want to use another access method, or engine, instead of the Version 8 engine, you can specify another engine name in the LIBNAME statement. For example, if you want to access Release 6.10 SAS data sets from your Version 8 SAS session, you can specify the V6 engine in the LIBNAME statement, as in the following example:

```
libname oldlib V6 'c:\sas610';
```

As another example, if you plan to share SAS files between Version 8 under Windows and Release 6 under Windows, you should use the V6 engine when assigning a libref to the SAS data library. Here is an example of specifying the V6 engine in a LIBNAME statement:

```
libname lib6 V6 'c:\sas6';
```

The V6 engine is particularly useful in your Version 8 SAS session if you are going to be accessing the same SAS files from a Release 6 SAS session. Remember that while Version 8 can read Release 6 SAS data sets, Release 6 cannot read Version 8 SAS data sets.

For more information about using engine names in the LIBNAME statement, see "Using SAS Files from Other Releases with Version 8 for Windows" on page 94 and

"Reading BMDP, OSIRIS and SPSS Files" on page 99. You can also refer to the LIBNAME statement in *SAS Language Reference: Dictionary*.

Using the LIBNAME Statement in SAS Autoexec Files

If you prefer, you can store LIBNAME statements in your SAS autoexec file so that the librefs are available as soon as the SAS System initializes. For example, your SAS autoexec file may contain the following statements:

```
libname test 'c:\mysasdir';
libname mylib ('c:\mydata','d:\tempdata');
libname oldlib V6 'c:\sas6';
```

For more information about how to create and use a SAS autoexec file, see "SAS Autoexec File" on page 14.

Assigning Multiple Librefs and Engines to a Folder

If a folder contains SAS files created by several different engines, only those SAS files created with the engine assigned to the given libref can be accessed using that libref. You can assign multiple librefs with different engines to a folder. For example, the following statements are valid:

```
libname one V6 'c:\mydir';
libname two V8 'c:\mydir';
```

Data sets referenced by the libref ONE are created and accessed using the compatibility engine (V6), whereas data sets referenced by the libref TWO are created and accessed using the default engine (V8). You can also have multiple librefs (using the same engine) for the same SAS data library. For example, the following two LIBNAME statements assign the librefs MYLIB and INLIB (both using the V8 engine) to the same SAS data library:

```
libname mylib V8 'c:\mydir\datasets';
libname inlib V8 'c:\mydir\datasets';
```

Because the engine name and SAS data library specifications are the same, the librefs MYLIB and INLIB are identical and can be used interchangeably.

Assigning SAS Libraries Using Environment Variables

You can also assign a libref to a SAS data library using environment variables instead of the LIBNAME statement or function. An *environment variable* equates one string to another within the Windows environment. The SAS System recognizes two kinds of environment variables: SAS environment variables and Windows environment variables. When you use a libref in a SAS statement, the SAS System first looks to see if that libref has been assigned to a SAS data library by a LIBNAME statement. If it has not, the SAS System searches first for any defined SAS environment variables and then for any Windows environment variables that match the specified libref. There are two ways of defining an environment variable to the SAS System:

- Use the SET system option. This defines a SAS (internal) environment variable.
- Issue a Windows SET command. This defines a Windows (external) environment variable. Alternatively under Windows NT, you can define environment variables using the System Properties dialog box accessed from the Control Panel, or by right-clicking on **My Computer** and selecting **Properties** from the pop-up menu.

CAUTION:
You cannot assign engines to environment variables. If you use environment variables as librefs, you must accept the default engine. △

The availability of environment variables makes it simple to assign resources to the SAS System prior to invocation.

SET System Option

You can use the SET system option to define a SAS environment variable. For example, if you store your permanent SAS data sets in the C:\SAS\MYSASDATA folder, you can use the following SET option in the SAS command or in your SAS configuration file to assign the environment variable TEST to this SAS data library:

```
-set test c:\sas\mysasdata
```

When you assign an environment variable, the SAS System does not resolve the environment reference until the environment variable name is actually used. For example, if the TEST environment variable is defined in your SAS configuration file, the environment variable TEST is not resolved until it is referenced by the SAS System. Therefore, if you make a mistake in your SET option specification, such as miss-typing a folder name, you do not receive an error message until you use the environment variable in a SAS statement.

Because Windows filenames can contain spaces or single quotes as part of their names, you should enclose the name of the physical path in double quotes when specifying the SET option. If you use the SET option in an OPTIONS statement, you must use quotation marks around the filename. For complete syntax of the SET system option, see “SET” on page 460.

Any environment variable name you use with a system option in your SAS configuration file must be defined as an environment variable before it is used. For example, the following SET option must appear before the SASUSER option that uses the environment variable TEST:

```
-set test "d:\mysasdir"
-sasuser !test
```

In the following example, environment variables are used with concatenated libraries:

```
-set dir1 "c:\sas\base\sashelp"
-set dir2 "d:\sas\stat\sashelp"
-sashelp (!dir1 !dir2)
```

Note that when you reference environment variables in your SAS configuration file or in a LIBNAME statement in your SAS programs, you must precede the environment variable name with an exclamation point (!).

It is recommended that you use the SET system option in your SAS configuration file if you invoke the SAS System through a Windows shortcut.

Windows SET Command

You can execute a Windows SET command prior to invoking the SAS System to create a Windows environment variable. You must define the environment variable prior to invoking the SAS System; you cannot define environment variables for SAS System use from a command prompt window from within a SAS session.

Operating Environment Information for Windows NT and Windows 95 Users: SAS can recognize environment variables only if they have been assigned in the same context that invokes the SAS session. That is, you must either define the environment variable in the Windows AUTOEXEC.BAT file that runs when Windows starts (thus creating a

global variable), or define the variable in a Command Prompt window from which you then start SAS. Under Windows NT, you can also define the environment variable using the System Properties dialog box accessed from the System item in the Control Panel.

If you define an environment variable in a Command Prompt window, and then start SAS from the Start menu (or another shortcut), SAS will not recognize the environment variable. △

The environment variables you define with the SET command can be used later within the SAS System as librefs. In the following example, the Windows SET command is used to define the environment variables PERM and BUDGET:

```
SET PERM="C:\MYSASDIR"
SET BUDGET="D:\SAS\BUDGET\DATA"
```

When you reference an external environment variable (one assigned with the Windows SET command) in your SAS programs (such as in a DATA or MERGE statement or in a SAS command), a note informing you the environment variable has been used is written to the SAS log. SAS does not recognize the environment variable as a libref until after you use it at least once during your SAS session, so the library name does not appear as a node in the SAS Explorer window until then.

Listing Libref Assignments

If you have assigned several librefs during a SAS session and need to refresh your memory as to which libref points where, you can use the SAS Explorer window, the LIBNAME command, or the LIBNAME statement to list all the assigned librefs.

If you are running the SAS System interactively, use the SAS Explorer window to view the active librefs. The SAS Explorer window lists all the librefs active for your current SAS session, along with the engine and the physical path for each libref. Any environment variables you have defined as librefs are listed, provided you have used them in your SAS session. If you have defined an environment variable as a libref but have not used it yet in a SAS program, the SAS Explorer window does not list it.

Note: You can use the LIBNAME command to invoke an 'Active Libraries' window which is a contents-only SAS Explorer that lists the active libraries, providing you with a quick view of your current libraries. △

You can use the following LIBNAME statement to write the active librefs to the SAS log:

```
libname _all_ list;
```

Clearing Librefs

You can clear a libref by using either the SAS Explorer window or a LIBNAME statement.

To clear a libref by using the SAS Explorer window, simply right-click on the node for the libref that you want to clear and select **Delete**. (For more information about using the SAS Explorer window to manage libraries, see *The Little SAS Book*.)

To clear a libref by using the LIBNAME statement, submit a LIBNAME statement using this syntax:

LIBNAME *libref*|_all_ <clear>;

If you specify a libref, only that libref is cleared. If you specify the keyword _all_, all the librefs you have assigned during your current SAS session are cleared. (SASUSER, SASHELP, and WORK remain assigned.)

Note: When you clear a libref defined by an environment variable, the variable remains defined, but it is no longer considered a libref—that is, it is not listed in the SAS Explorer window. You can use the variable in another LIBNAME statement to create a new libref. △

The SAS System automatically clears the association between librefs and their respective data libraries at the end of your job or session. If you want to associate the libref with a different SAS data library during the current session, you do not have to end the session or clear the libref. The SAS System automatically reassigns the libref when you use it to name a new library from within either the SAS Explorer window or a LIBNAME statement.

Understanding How Concatenated SAS Data Libraries Are Accessed

When you use the concatenation feature to specify more than one physical folder for a libref, the SAS System uses the following protocol for determining which folder is accessed. (The protocol illustrated by these examples applies to all SAS statements and procedures that access SAS files, such as the DATA, UPDATE, and MODIFY statements in the DATA step and the SQL and APPEND procedures.)

Input and Update Access

When a SAS file is accessed for input or update, the first SAS file found by that name is the one accessed. For example, if you submit the following statements and the file OLD.SPECIES exists in both folders, the one in the C:\MYSASDIR folder is printed:

```
libname old ('c:\mysasdir','d:\saslib');
proc print data=old.species;
run;
```

The same would be true if you opened OLD.SPECIES for update with the FSEDIT procedure.

Output Access

If the data set is accessed for output, it is always written to the first folder, provided that the folder exists. If the folder does not exist, an error message is displayed. For example, if you submit the following statements, the SAS System writes the OLD.SPECIES data set to the first folder (C:\MYSASDIR), replacing any existing data set with the same name:

```
libname old ('c:\mysasdir','d:\saslib');
data old.species;
   x=1;
   y=2;
run;
```

If a copy of the OLD.SPECIES data set exists in the second folder, it is not replaced.

Accessing Data Sets with the Same Name

One possibly confusing case involving the access protocols for SAS files occurs when you use the DATA and SET statements to access data sets with the same name. For example, suppose you submit the following statements and TEST.SPECIES originally exists only in the second folder, D:\MYSASDIR:

```
libname test ('c:\sas','d:\mysasdir');
data test.species;
```

```
   set test.species;
   if value1='y' then
      value2=3;
run;
```

In this case, the DATA statement opens TEST.SPECIES for output according to the output rules; that is, the SAS System opens a data set in the first of the concatenated libraries (C:\SAS). The SET statement opens the existing TEST.SPECIES data set in the second (D:\MYSASDIR) folder, according to the input rules. Therefore, the original TEST.SPECIES data set is not updated; rather, two TEST.SPECIES data sets exist, one in each folder.

Using the SASUSER Data Library

The SAS System automatically creates a SAS data library with the libref SASUSER. This library contains, among other SAS files, your user profile catalog. By default under Windows, the SASUSER libref points to a folder called "My SAS Files\V8", located under the working folder of your current SAS session. If a SASUSER folder does not exist, the SAS System creates one. You can use the SASUSER system option to make the SASUSER libref point to a different SAS data library. For more information about your profile catalog, see "Profile Catalog" on page 15. For more information about the SASUSER system option, see "SASUSER" on page 459. The SAS System stores other files besides the profile catalog in the SASUSER folder. For example, the sample data sets provided with SAS/ASSIST software are stored in this folder.

CAUTION:

You cannot change the engine associated with the SASUSER data library. The SASUSER data library is always associated with the V8 engine. If you try to assign another engine to this data library, you receive an error message. Therefore, even if you have set the ENGINE system option to another engine, any SAS files that are created in the SASUSER data library are Version 8 SAS files. △

Using the WORK Data Library

The WORK data library is the storage place for temporary SAS files. By default under Windows, the WORK data library is created as a subfolder of the SASWORK folder. This subfolder is named #TD*nnnnn*, as discussed in "WORK Data Library" on page 16. Temporary SAS files are available only for the duration of the SAS session in which they are created. At the end of that session, they are deleted automatically. By default, any file that is not assigned a two-level name is automatically considered to be a temporary file. A special libref of WORK is automatically assigned to any temporary SAS data sets created. For example, if you run the following SAS DATA step to create the data set SPORTS, a temporary data set named WORK.SPORTS is created:

```
data sports;
   input @1 sport $10. @12 event $20.;
   cards;
volleyball co-recreational
swimming   100-meter freestyle
soccer     team
;
```

If you display the SAS Explorer window now, you will see the SPORTS data set in the WORK folder.

You can display all the temporary data sets that are created during this session from the SAS Explorer window by double-clicking the Libraries folder icon and then double-clicking the Work folder icon. From the Active Libraries (LIBNAME) window, double-click the Work folder icon to see the temporary data sets.

CAUTION:
You cannot change the engine associated with the WORK data library. The WORK data library is always associated with the V8 engine. If you try to assign another engine to this data library, you receive an error message. Therefore, even if you have set the ENGINE system option to a different engine, any SAS files that are created in the WORK data library are Version 8 SAS files. △

Using an Environment Variable

You can use an environment variable in your WORK data library specification, similar to the method illustrated earlier with the SASUSER system option. Use this technique when you do not want to use the default location for your WORK data library. You can put something similar to the following in your SAS configuration file to set up an environment variable to use for your WORK data library:

```
-set myvar c:\ tempdir
-work !myvar use
```

The SET option associates the MYVAR environment variable with the C:\TEMPDIR folder. Then the WORK option tells the SAS System to use that folder for the SAS WORK data library. The USE suboption in the WORK option tells the SAS System to store temporary SAS files in the folder specified by the WORK option. If you do not specify the USE suboption, by default the SAS System creates a subfolder beneath the specified folder. This subfolder is named #TD*nnnnn*, where *nnnnn* is a unique number associated with your SAS session. When you exit your SAS session, these folders and any files they contain are removed.

Note: If you specify the USE suboption in the default configuration file, you will not be able to run multiple SAS sessions on the same machine, since each SAS session requires a unique WORK folder.
Also, do not assign an environment variable named TEMP or TMP, as these variable names are usually already used by Windows. △

Using the USER Libref

Although by default SAS files with one-level names are temporary and are deleted at the end of your SAS session, you can use the USER libref to cause SAS files with one-level names to be stored in a permanent SAS data library. For example, the following statement causes all SAS files with one-level names to be permanently stored in the C:\MYSASDIR folder:

```
libname user 'c:\mysasdir';
```

When you set the USER libref to a folder as in the previous example, if you want to create or access a temporary data set you must specify a two-level name for the data set with WORK as the libref. You can also assign the USER libref when you invoke the SAS System by using the USER system option. For more information about the USER system option, refer to “USER” on page 472 and *SAS Language Reference: Dictionary*.

Note: You can assign other engines to the USER libref if you want the data sets that are saved with one-level names to be stored in a format for use with other releases of the SAS System. △

Accessing SAS Files from Multiple SAS Sessions

If you are running multiple SAS sessions, whether on a single machine or across a network, you can have multiple access to the same SAS file when you are reading from it.

If you have SAS/SHARE installed, the VIEWTABLE window and the FSEDIT or FSVIEW windows allow mulitple users to edit the same SAS file. When you edit a data set using the VIEWTABLE window, you can set the editing mode to either Table Level Edit Access or Row Level Edit Access. When you select Table Level Edit Access, only you have access to the data set. Row Level Edit access allows multiple users to access the same SAS file, but only one user can access and make changes to a single record (observation) at a time.

To open a data set in the VIEWTABLE window, from the SAS Explorer window:

1. double-click the Libraries icon
2. double-click the library containing the data set
3. double-click the data set.

To edit the data set, select

Edit ▶ Edit Mode

and then select either **Table Level Edit** or **Row Level Edit**.

When you edit a data set using FSEDIT or FSVIEW, you can set the update mode to either MEMBER or RECORD. When you select MEMBER mode, only you have access to the data set. When you select RECORD mode, multiple users can write to the same SAS file but only one user can update a single record (observation) at a time.

To open a data set using FSEDIT or FSVIEW:

1. type FSEDIT or FSVIEW in the command bar
2. double-click the library name in the Select a Member dialog box
3. double-click the data set name.

To edit the data set, select

Edit ▶ Update...

and then select either the **MEMBER** or **RECORD** radio button.

The RSASUSER system option, described in "RSASUSER" on page 452 allows you to share the SASUSER data library. If multiple users need update access to common SAS data sets, use SAS/SHARE software.

For details about rules for multiple user access to the same data set and its members, see the SAS System Help and *SAS/SHARE User's Guide*.

Using SAS Files from Other Releases with Version 8 for Windows

This section discusses using SAS files from previous releases of SAS for Windows. For information on transporting files from other operating environments to the Windows environment, see *Moving and Accessing SAS Files across Operating Environments*.

If you intend to use pre-Version 8 SAS files with only the SAS System for Version 8, you will see improved performance if you convert your pre-Version 8 files to the Version 8 files.

Note: If you will be using your SAS files with previous releases of the SAS System, do not convert your files to Version 8 files. △

Table 3.4 on page 95 summarizes the steps you must complete to use your existing SAS data sets and catalogs for Windows with Version 8.

Table 3.4 Migrating Existing SAS Files to Version 8

Release	SAS Data Sets	SAS Catalogs
6.12 for Windows	transparent	CPORT and CIMPORT
6.11 for Windows	transparent	CPORT and CIMPORT
6.10 for Windows	transparent	CPORT and CIMPORT
6.09 for Windows NT	transparent	CPORT and CIMPORT
6.08 for Windows	transparent	CPORT (C16PORT) and CIMPORT
6.04 and 6.03 for DOS	V604	CPORT and CIMPORT

As the table shows, except for Release 6.04 and Release 6.03 data sets, Release 6 data sets do not need to be converted to Version 8 data sets for Version 8 to read the data sets. All Release 5 and Release 6 SAS catalogs must be transported to be used in Version 8.

Note: Version 8 of SAS under Windows does not read OS/2 SAS data sets directly as it did in Release 6. To use OS/2 SAS data sets in the Windows operating environment, you must either transport them or use Cross Environment Data Access (CEDA). For more information, see *Moving and Accessing SAS Files across Operating Environments*. △

Using Release 6 SAS Data Sets in Version 8

This section discusses using SAS data sets created using Release 6.03 through Release 6.12.

Using Release 6.08 Through Release 6.12 Data Sets

If your SAS library contains only Release 6 SAS files, the SAS System for Version 8 automatically determines the appropriate engine to use for Version 6.08 and later SAS data sets. If your SAS files are in a mixed mode library that possibly contains SAS data sets from Release 6 and Version 8, you must specify the *engine* parameter on the LIBNAME statement or the engine will default to V8. For information on using Release 6.03 and Release 6.04 SAS files, see “Using Release 6.03 and Release 6.04 SAS Data Sets” on page 96.

For example, knowing that the ’c:\mydata’ SAS library contains only Release 6 files, the following SAS statements print a Release 6 SAS data set named WINDATA.SALEFIGS created under Version 3.1.1 of Windows:

```
libname windata 'c:\mydata';
proc print data=windata.salefigs;
   title 'Sales Figures';
run;
```

Where all SAS files in the library are Release 6 SAS files, you can omit the engine parameter because SAS automatically detects the V6 engine.

Using the same example, if you are unsure or if you know that the SAS library is a mixed mode library, you must specify the engine name in the LIBNAME statement to access the V6 files:

```
libname windata v6 'c:\mydata';
proc print data=windata.salefigs;
   title 'Sales Figures';
run;
```

You may want to convert your Release 6 SAS files to Version 8 format. There are separate considerations for converting SAS data sets and SAS catalogs.

Using Release 6.03 and Release 6.04 SAS Data Sets

The V604 engine enables you to read from and write to Release 6.03 and Release 6.04 SAS data sets directly from your Version 8 SAS session. (Remember that there is no difference between Release 6.04 and Release 6.03 SAS data sets.) This feature is useful when you have SAS data sets that you want to share between Release 6.04 for PCs and Version 8 under Windows. The V604 engine is supported only for SAS data sets (member type DATA). For example, if you have a Release 6.04 SAS data set named MYLIB.FRUIT that you want to print, you can submit the following statements from a Version 8 session:

```
libname mylib v604 'c:\sas604';
proc print data=mylib.fruit;
run;
```

While it is useful to be able to access SAS data sets created in a previous release of the SAS System, you cannot take advantage of the full functionality of the Version 8 SAS System, which includes creating indexes and database views, unless you convert the data sets reated with earlier releases of the SAS System to Version 8 data sets.

Converting Release 6.08 Through Release 6.12 SAS Data Sets

While access to Release 6 SAS data sets is quite easy when you use the V6 engine, you may want to consider converting your SAS data sets to Version 8 format if you access them often and do not need to read the files from Release 6 any more. The data set format of Version 8 of the SAS System is more efficient than the Release 6 format and there are new Version 8 features that cannot be used unless the data sets are converted.

The following SAS statements use the COPY procedure to convert all the Release 6 SAS data sets in the V6DATA SAS data library to Version 8 and to format and store the new data sets in the WINDATA library:

```
libname v6data v6 'c:\mydata';
libname windata V8 'd:\newdata';
proc copy in=v6data out=windata memtype=data;
run;
```

Alternatively, you can use the DATA step to do the conversion, as in the following example. This technique works well if you want to convert only one or two data sets in a particular SAS data library.

```
libname v604data v604 'c:\mydata';
libname windata V8 'd:\newdata';
data windata.eggplant;
   set v604data.eggplant;
run;
```

You can also use the CPORT and CIMPORT procedures to perform the conversion. Remember that when you convert a data set to Version 8 format, you can no longer read that data set from Release 6. For information on using the CPORT and CIMPORT procedures, see *Moving and Accessing SAS Files across Operating Environments*and *SAS Procedures Guide*.

Note: Do not convert your Release 6. files to Version 8 if you need to access the files from both releases. △

Creating Release 6 Data Sets

You may need to create Release 6 SAS data sets from your Windows SAS session. This is similar to reading Release 6 data sets in that you use the V6 engine. For example, the following SAS statements use the V6 engine to create a SAS data set named QTR1. The raw data are read from the external file associated with the fileref MYFILE.

```
libname windata v6 'c:\mydata';
filename myfile 'c:\qtr1data.dat';
data windata.qtr1;
   infile myfile;
   input saledate amount;
run;
```

Converting Release 6 SAS Catalogs in Version 8

Because of the differences in the internal structures of the operating systems, you must use the CPORT and CIMPORT procedures to convert Release 6 SAS catalogs created under Windows to Version 8 format before you can use the catalogs in your Version 8 SAS session under Windows. Here are the basic steps to follow:

1. Using the CPORT procedure in your Release 6 SAS session, create a transport file containing the SAS catalog to be converted.
2. Transfer the file (perhaps on a network or diskette) to a place your Version 8 SAS session can read it.
3. Use the CIMPORT procedure from your Version 8 SAS session to read the transport file and create a converted SAS catalog.

For information on using the CPORT and CIMPORT procedures, see *Moving and Accessing SAS Files across Operating Environments* and *SAS Procedures Guide*.

Converting Release 6.08 SAS Catalogs

If you are converting directly from Release 6.08 to Version 8, you can use the CPORT procedure in Release 6.08 to create a transport file, and then use the Version 8 CIMPORT procedure to convert the catalog to a Version 8 catalog. However, the

HSERVICE and TOOLBOX catalog entries are not portable if you use CPORT from a Release 6.08 session.

An alternative way to convert Release 6.08 catalogs is to use the C16PORT procedure provided in Release 6.10 through Release 6.12. SAS provided the C16PORT procedure to convert the 16-bit catalogs created with Release 6.08 under Windows to a 32-bit format that the SAS System can use. You can use the C16PORT procedure from within one of these earlier releases of the SAS System to create a catalog that can later be read by Version 8. (The C16PORT procedure is not available in Version 8.)

Here are the steps to follow to convert your SAS catalogs from Release 6.08 under Windows to Version 8:

1. While in your Release 6.10, Release 6.11, or Release 6.12 session, use the C16PORT procedure (described in the documentation for those releases) to create a transport file containing the SAS catalog from Release 6.08.
2. Use the CIMPORT procedure from your Release 6.10, Release 6.11, or Release 6.12 SAS session to read the transport file and create a converted SAS catalog.
3. Use a Version 8 SAS session to access the converted catalog.

If you want to convert a catalog that currently exists on another machine running Release 6.08 for Windows, you must first transfer the file (perhaps on a network or a diskette) to a place where your Version 8 session can read it.

The following example uses the C16PORT procedure in Release 6.12 to create a transport file from the INLIB.CAT catalog, then creates a Release 6.12 catalog (OUTLIB.CAT) using the CIMPORT procedure.

```
   /* Folder where catalog    */
   /* 'cat.sc2' resides       */
libname inlib 'c:\cat608';
   /* Folder where catalog    */
   /* 'cat.sc8' will reside   */
libname outlib 'c:\cat612';
proc c16port file='transprt' c=inlib.cat;
run;
proc cimport infile='transprt' c=outlib.cat;
run;
```

The Release 6.12 SAS catalog can now be read by Version 8. For information on the CPORT and CIMPORT procedures, see *SAS Procedures Guide* and *Moving and Accessing SAS Files across Operating Environments*.

Converting Release 6.03 and Release 6.04 SAS Catalogs to Version 8

If you want to convert Release 6.04 SAS catalogs to their Version 8 counterparts, see *SAS Technical Report P-195, Transporting SAS Files between Host Systems*.

Using Version 8 SAS Files with Previous Releases

It is possible to move SAS files between your Version 8 SAS session under Windows and Release 6.08 through Release 6.12 of the SAS System under Windows using the CPORT and CIMPORT procedures.

Note: When you transport Version 8 files to previous releases, Version 8 features will not be available from the transport file △

For information about PROC CPORT, PROC CIMPORT, and about back-porting SAS/AF applications, see *SAS Procedures Guide*. For more information about

back-porting SAS/EIS applications, see the SAS online Help for SAS/EIS. For more information about transporting files in general, see *Moving and Accessing SAS Files across Operating Environments*.

Using Version 5 and Other Remote Host SAS Files in Version 8

You can directly access remote host SAS data sets, including Version 5, Release 6.07, and Release 6.08 SAS data sets created on a remote host. For example, you may have a Version 5 SAS data set on VSE or a Release 6.07 data set on OS/390. The following explanation uses Version 5, but also applies to these other releases.

To directly access a Version 5 SAS data set created on a remote host computer, use SAS/CONNECT software to establish a communication link between your local SAS session under Windows and a remote SAS session under the other host. Once you have established a link, you can remote submit SAS statements to process the Version 5 data on the remote host computer, or you can download the Version 5 data set to a Version 8 data set under Windows. SAS/CONNECT software provides a convenient one-step way to transport SAS files from system to system. For more information about downloading files, see *SAS/CONNECT User's Guide*.

As an alternative, you can create a transport data set from your Version 5 data set and transport your file from the host to Windows. Then you can use the XPORT engine to create a Version 8 file from this transport file. If you are accessing Version 6 remote SAS data sets, you can also use PROC CPORT to create the transport file.

The method for transporting data sets differs from the method for transporting SAS catalogs. For more information about transporting SAS files, see *SAS Technical Report P-195, Transporting SAS Files between Host Systems*.

Reading BMDP, OSIRIS and SPSS Files

Version 8 of the SAS System provides three interface library engines that enable you to access external data files directly from a SAS program: the BMDP, OSIRIS and SPSS engines. These engines are all read-only. Because they are sequential engines (that is, they do not support random access of data), these engines cannot be used with the POINT= option in the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedures. You can use PROC COPY or a DATA step to copy the system file to a SAS data set and then perform these functions on the SAS data set. Also, because they are sequential engines, some procedures (such as the PRINT procedure) give a warning message that the engine is sequential. With these engines, the physical filename associated with a libref is an actual filename, not a folder. This is an exception to the rules concerning librefs.

You can also use the CONVERT procedure to convert BMDP, OSIRIS and SPSS files to SAS data files. For more information, see "CONVERT" on page 358.

BMDP Engine

The BMDP interface library engine enables you to read BMDP DOS files from the BMDP statistical software package directly from a SAS program. The BMDP engine is a read-only engine. The following discussion assumes you are familiar with the BMDP save file terminology.

To read a BMDP save file, you must issue a LIBNAME statement that explicitly specifies you want to use the BMDP engine. In this case, the LIBNAME statement take the following form:

LIBNAME *libref* BMDP <'filename'>;

In this form of the LIBNAME statement, *libref* is a SAS libref and *filename* is the BMDP physical filename. If the libref appears previously as a fileref, you can omit *filename* because the physical filename associated with the fileref is used. This engine can only read BMDP save files created under DOS.

Because there can be multiple save files in a single physical file, you reference the CODE= value as the member name of the data set within the SAS language. For example, if the save file contains CODE=ABC and CODE=DEF and the libref is MYLIB, you reference them as MYLIB.ABC and MYLIB.DEF. All CONTENT types are treated the same, so even if member DEF is CONTENT=CORR, it is treated as CONTENT=DATA.

If you know that you want to access the first save file in the physical file, or if there is only one save file, you can refer to the member name as _FIRST_. This is convenient if you don't know the CODE= value.

BMDP Engine Examples

In the following example, the physical file MYBMDP.DAT contains the save file ABC. This example associates the libref MYLIB with the BMDP physical file, then runs the CONTENTS and PRINT procedures on the save file:

```
libname mylib bmdp 'mybmdp.dat';
proc contents data=mylib.abc;
run;
proc print data=mylib.abc;
run;
```

The following example uses the LIBNAME statement to associate the libref MYLIB2 with the BMDP physical file. Then it prints the data for the first save file in the physical file:

```
libname mylib2 bmdp 'mybmdp.dat';
proc print dat=mylib2._first_;
run;
```

OSIRIS Engine

Because the Inter-University Consortium on Policy and Social Research (ICPSR) uses the OSIRIS file format for distribution of its data files, the SAS System provides the OSIRIS interface library engine to support the many users of the data from ICPSR and to be compatible with PROC CONVERT, which is described in "CONVERT" on page 358.

The read-only OSIRIS engine enables you to read OSIRIS data and dictionary files directly from a SAS program. These files must be stored in EBCDIC format. This means you must have downloaded the OSIRIS files from your host computer in binary format. The following discussion assumes you are familiar with the OSIRIS file terminology. *

To read an OSIRIS file, you must issue a LIBNAME statement that explicitly specifies you want to use the OSIRIS engine. In this case, the LIBNAME statement takes the following form:

* See documentation provided by the Institute for Social Research for more information.

LIBNAME *libref* OSIRIS '*data-filename*' DICT='*dictionary-filename*';

In this form of the LIBNAME statement, *libref* is a SAS libref, *data-filename* is the physical filename of the OSIRIS data file, and *dictionary-filename* is the physical filename of the OSIRIS dictionary file. The *dictionary-filename* argument can also be an environment variable name or a fileref. (Do not put it in quotes if it is an environment variable name or fileref.) The DICT= option must appear because the engine requires both files.

OSIRIS data files do not have member names. Therefore, you can use whatever member name you like. You can use the same OSIRIS dictionary file with different OSIRIS data files. Simply write a separate LIBNAME statement for each one.

The layout of an OSIRIS data dictionary is consistent across operating systems. This is because the OSIRIS software does not run outside of the OS/390 environment, but the engine is designed to accept an OS/390 data dictionary on any other operating system under which the SAS System runs. It is important that the OSIRIS dictionary and data files not be converted from EBCDIC to ASCII; the engine expects EBCDIC data. There is no specific file layout for the OSIRIS data file. The file layout is dictated by the contents of the OSIRIS dictionary file.

OSIRIS Engine Example

In the following example, the data file is MYOSIRIS.DAT, and the dictionary file is MYOSIRIS.DIC. The example associates the libref MYLIB with the OSIRIS files and then runs PROC CONTENTS and PROC PRINT on the data:

```
libname mylib osiris 'myosiris.dat'
        dict='myosiris.dic';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

SPSS Engine

The SPSS interface library engine enables you to read SPSS/PC files directly from a SAS program. This is a read-only engine. The following discussion assumes you are familiar with the SPSS save file terminology. **

To read an SPSS save file, you must issue a LIBNAME statement that explicitly specifies you want to use the SPSS engine. In this case, the LIBNAME statement takes the following form:

LIBNAME *libref* SPSS <'*filename*'>;

In this form of the LIBNAME statement, the *libref* argument is a SAS libref, and *filename* is the SPSS physical filename. If the libref appears also as a fileref, you can omit *filename* because the physical filename associated with the fileref is used. The SPSS/PC native file format, as well as the export file format, is supported. The engine determines which format is used and reads the save file accordingly. Native files must be created under PC DOS. Export files can originate from any operating system.

Because SPSS/PC files do not have internal names, you can refer to them by any member name you like. (The example in this discussion uses _FIRST_ .)

** See documentation provided by SPSS Inc. for more information.

SPSS Engine Example

The following example associates the libref MYLIB with the physical file MYSPSS.DAT in order to run PROC CONTENTS and PROC PRINT on the save file:

```
libname mylib spss 'myspss.dat';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

Transferring SAS Files between Operating Systems

For complete information on transferring SAS files between operating systems, see *Moving and Accessing SAS Files across Operating Environments*.

Accessing Database Files with SAS/ACCESS Software

SAS/ACCESS software provides an interface between the SAS System and several database management systems (DBMS) that run under Windows. The interface consists of three procedures and an interface view engine, which can perform the following tasks:

ACCESS procedure
: creates SAS/ACCESS descriptor files that describe DBMS data to the SAS System. It also can create a SAS data file from DBMS data.

DBLOAD procedure
: loads SAS or other DBMS data into a DBMS file.

SQL Procedure Pass-Through Facility
: accesses data from several different relational DBMSs, including ODBC and SYBASE.

interface view engine
: enables you to use descriptor files in SAS programs to access DBMS data directly and enables you to specify descriptor files in SAS programs to update, insert, or delete DBMS data directly.

For more information about using SAS/ACCESS software under Windows, consult *SAS/ACCESS Software for PC File Formats: Reference* and other available SAS/ACCESS documentation.

Using the SAS ODBC Driver to Access SAS Data from Other Applications

The SAS ODBC driver is an implementation of the open database connectivity (ODBC) standard that enables you to access, manipulate, and update SAS data sources. These data sources can include SAS data sets, flat files, VSAM files, as well as data from any database management system (DBMS) for which you have licensed SAS/ACCESS software. For information about how to access data from other Windows applications that comply with the ODBC standard, see the SAS online Help.

The SAS ODBC Driver accesses data by communicating with either a local or remote (SAS/SHARE) SAS server session using the TCP/IP protocol. The TCP/IP protocol

enables users to access remote SAS servers on a variety of host platforms. A SAS server is a SAS procedure (either PROC SERVER or PROC ODBCSERV) that runs in its own SAS session; it accepts input and output requests from other SAS sessions and from the SAS ODBC driver on behalf of the ODBC-compliant application. For remote access to SAS data, a SAS server must be installed on the server machine, but not on the client machine.

The SAS ODBC Driver is included with base SAS. Remote server configurations that use the SAS ODBC driver require that these SAS products be installed:

- Base SAS
- SAS/SHARE
- SAS/SHARE*NET

For details about installing and configuring the SAS ODBC Driver, see the installation documentation for the SAS System under Windows. For more information on configuring and using the SAS ODBC Driver, see *SAS ODBC Driver User's Guide and Programmer's Reference*.

Using the SAS UODBC Driver to Access SAS Data from Other Applications

The SAS Universal ODBC (UODBC) driver is an implementation of the open database connectivity (ODBC) standard that enables you to obtain read-only access to SAS data sources. Data sources include SAS data sets, catalogs, consolidation databases, multi-dimensional databases, and JMP data.

The SAS UODBC driver does not require the SAS System in order to access SAS data. It is useful when read-only access to SAS data is required, but the power of the data manipulation capabilities of the SAS ODBC driver is not needed. A typical use of UODBC is on an Internet WEB page for sharing read-only real-time data.

For information about how to access data from other Windows applications that comply with the UODBC standard, see the SAS online Help.

The SAS UODBC driver supports these data access methods:

Direct
: enables the driver to be configured to open a data source by accessing the file that is either on the driver's local machine or on a network drive that is attached to the driver's local machine.

FTP
: enables users to access SAS data files that are stored on a File Transfer Protocol (FTP) server. The UODBC driver can be configured to log in to the server, if necessary.

HTTP
: enables users to access SAS data files that are stored on a WEB server that supports the Hypertext Transfer Protocol (HTTP). The UODBC driver can be configured to log in to the server, if necessary.

The SAS UODBC driver is not included as part of base SAS , but it is available as a separate product. For evaluation purposes, a trial version of the driver may be downloaded from SAS Institute at **`http://www.sas.com`**.

For details about installing the SAS UODBC Driver, see the installation documentation for the SAS System under Windows.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. pp.555.

SAS Companion for the Microsoft Windows Environment, Version 8

ISBN 1–58025–524–8

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

1st printing, September 1999

The Institute is a private company devoted to the support and further development of its software and related services.