CHAPTER

6

Using SAS Engines

Introduction

Chapter 5, “Using SAS Files,” on page 123 describes how to assign a libref to a SAS data library. It explains how engine names can be associated with librefs to create and access SAS files of various types. The engines that are available under Version 8 enable you to create, read, write, and, in most cases, update files of different types in different formats.

This section provides more detailed information about the engines that are available in the OpenVMS operating environment.

Engines Available under OpenVMS

Table 6.1 on page 141 lists the engines that are available under OpenVMS in Version 8 and tells you where to look for more information about each engine. The preferred name of each engine is listed first, followed by acceptable nicknames, if any.

Table 6.1 SAS Engines under OpenVMS

Engine Name (Alias)	Description	See...
V8 (BASE)	accesses Version 8 SAS files on disk	"Using the V8 Engine" on page 141
V8TAPE (TAPE)	accesses Version 8 sequential- format SAS files	"Using the V8TAPE Engine" on page 144
V7	accesses Version 7 SAS files on disk	"Using the V7 ENGINE" on page 145
V7TAPE	accesses Version 7 sequential-format SAS files	"Using the V7TAPE Engine" on page 148
V6	accesses Version 6 SAS data files on disk	"Using the V6 Engine" on page 149
V6TAPE	accesses Version 6 sequential-format SAS files	"Using the V6TAPE Engine" on page 151
CONCUR	provides concurrent update access to SAS data sets	"Using the CONCUR Engine" on page 154
V5	accesses Version 5 SAS data sets on disk	"Using the V5 Engine" on page 153
INGRES	accesses CA-OpenIngres database files	"Using the DBMS Interface Engines" on page 161
ORACLE	accesses ORACLE database files	"Using the DBMS Interface Engines" on page 161
RDB	accesses Oracle Rdb database files	"Using the DBMS Interface Engines" on page 161
SQLVIEW	accesses data views that are described by the SQL procedure	*SAS Guide to the SQL Procedure: Usage and Reference*
SYBASE	accesses SYBASE database files	"Using the DBMS Interface Engines" on page 161
XPORT	accesses transport files	*Moving and Accessing SAS Files across Operating Environments*
OSIRIS	provides read-only access to OSIRIS files	"Using the OSIRIS and SPSS Engines" on page 161
SPSS	provides read-only access to SPSS files	"Using the OSIRIS and SPSS Engines" on page 161

File Types Created by Each Engine

For information about the OpenVMS file types that the SAS System uses for SAS files, see "OpenVMS File Types Used by the SAS System" on page 10.

Using the V8 Engine

The section describes how to access Version 8 data libraries using the default engine, V8. The V8 engine supports indexing and compression of observations.

When to Use the V8 Engine

Generally, SAS automatically determines the appropriate engine to use for accessing the files in the library. If you want to create a new library with an engine other than the default engine, you can override the automatic selection.

You use the V8 engine to create SAS data libraries on disk and to read from, write to, or update those libraries. The V8 engine is the default engine for new SAS Version 8 data libraries, unless the default engine has been changed with the ENGINE= system option.

How to Select the V8 Engine

There are three ways to select the V8 engine:

- Specify V8 as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify V8TAPE in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.

  Note: Use BASE as the engine name if you write programs that create new SAS data libraries and you want to create the data libraries in the latest available format. In Version 8, BASE is an alias for the V8 engine, and it will be an alias for newer engines in subsequent releases. △

- For an existing SAS data library on disk that contains only Version 8 SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then selects the V8 engine automatically. SAS also selects the V8 engine automatically if you use the DCL DEFINE command to assign an OpenVMS logical name to an existing Version 8 SAS data library on disk and then use that logical name as a libref in a SAS file specification.
- Set the value of the ENGINE= system option to V8. This option tells SAS which engine to use as the default when no engine is specified and there are no existing data sets, or when the directory is in mixed mode.

Member Types Supported

The V8 engine supports files with all Version 8 member types:

ACCESS	ITEMSTOR
AUDIT	MDDB
BACKUP	PROGRAM
CATALOG	PUTILITY
DATA	SASODS
DMDB	UTILITY

FDB VIEW

INDEX

Engine/Host Options for the V8 Engine

The V8 engine provides several engine/host options that control the creating and access of SAS data sets. Most of the following options correspond to options that are available through OpenVMS Record Management Services (RMS).

You can use the following engine/host options with the V8 engine:

ALQ=
specifies the number of OpenVMS disk blocks to allocate initially to a data set when it is created. The value can range from 0 to 2,147,483,647. If the value is 0, the minimum number of blocks that is required for a sequential file is used. OpenVMS RMS always rounds the value up to the next disk cluster boundary.

The ALQ= option (allocation quantity) corresponds to the FAB$L_ALQ field in OpenVMS RMS. For additional details, see the data set option "ALQ=" on page 249 and *Guide to OpenVMS File Applications*.

ALQMULT=
specifies the number of pages that are preallocated to a file when it is created. The value can range from 1 to 128. The default value is 10. The ALQMULT= option is related to the ALQ= option.

For additional details, see the data set option "ALQMULT=" on page 250.

BUFSIZE=
specifies the size of the internal I/O buffer used for output data sets. The value can range from 0 to the maximum allowed under OpenVMS. The engine tries to use a value in the range of 8,192 to 32,768 if possible. The BUFSIZE= option must be specified in increments of 512.

For additional details, see the data set option "BUFSIZE=" on page 252.

CACHENUM=
specifies the number of I/O data caches used per SAS file. The value can range from 1 to 16. The default value is 10. The CACHENUM= option is used in conjunction with the CACHESIZ= option.

For additional details, see the data set option "CACHENUM=" on page 253.

CACHESIZ=
controls the size (in bytes) of the data cache used to buffer I/O pages. The value can range from 0 to 65,024. By default, the chosen value is an even multiple of the file's page size. A value of 0 specifies to use no data cache. Memory is consumed for the data cache, and multiple caches may be used for each data set opened. The disadvantage of large CACHESIZ= values is large consumption of memory. The advantage of large CACHESIZ= values is a reduction in the number of I/Os required to read from or write to a file.

The CACHESIZ= and BUFSIZE= options are similar, but they have important differences. The BUFSIZE= option specifies the file's page size, which is permanent. It can only be set on file creation. The CACHESIZ= option is the size of the internal memory cache used for the life of the current open, so it can change any time the file is opened. Also, BUFSIZE= cannot be used as an engine/host option; it is only valid as a data set option.

For additional details, see the data set option "CACHESIZ=" on page 253.

CNTLLEV=
specifies the level of shared access allowed to SAS data sets. Users can be given both read and write access to a data set that is opened for input only. By default, only shared read access is allowed.
For additional details, see the data set option "CNTLLEV=" on page 255.

DEQ=
specifies the number of OpenVMS disk blocks to add each time OpenVMS RMS automatically extends a data set during a write operation. The value can range from 0 to 65,535. OpenVMS RMS always rounds the value up to the next disk cluster boundary. A large value can result in fewer file extensions over the life of the file; a small value results in numerous file extensions over the life of the file. A file with numerous file extensions that may be noncontiguous slows record access.
If the value specified is 0, OpenVMS RMS uses the default value for the process.
The DEQ= option (default file extension quantity) corresponds to the FAB$W_DEQ field in OpenVMS RMS. For additional details, see the data set option "DEQ=" on page 256 and *Guide to OpenVMS File Applications*.

DEQMULT=
specifies the number of pages to extend a SAS file. The value can range from 1 to 128. The default value is 5.
For additional details, see the data set option "DEQMULT=" on page 257.

Using the V8TAPE Engine

In contrast to the V8 engine, the V8TAPE engine does not support indexing and compression of observations. Also, you cannot have a mixed mode, sequential-format SAS data library; you can store only Version 8, Version 7, or Version 6 sequential SAS data sets on one tape, not some of each.

Note: For information about accessing SAS files on tape, see "Accessing SAS Files on Tape" on page 134. △

When to Use the V8TAPE Engine

Use the V8TAPE engine to create Version 8 sequential-format SAS data libraries either on disk or on tape and to access files in Version 8 sequential data libraries. The primary purpose of this engine is to enable you to back up Version 8 SAS data sets, catalogs, or whole data libraries. The V8TAPE engine makes it possible to back up applications that contain both SAS data sets and SAS catalogs.

In contrast to the V8 engine, the V8TAPE engine has the following limitations:

- Because the V8TAPE engine is a sequential engine, it cannot be used with the POINT= option of the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedure. If you want to use these features of the SAS language with sequential SAS data sets, then use the COPY procedure to copy them to a disk-format SAS data library. (See "Notes on Tape Usage" on page 136.)
- In a single DATA step or PROC step, you can use only one SAS data set from a particular sequential SAS data library.

How to Select the V8TAPE Engine

There are three ways to select the V8TAPE engine:

- Specify V8TAPE as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify V8TAPE in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.

  Note: Use TAPE as the engine name if you write programs that create new SAS data libraries and you want to create the data libraries in the latest available format. In Version 8, TAPE is an alias for the V8TAPE engine, and it will be an alias for newer sequential engines in subsequent releases. △
- For an existing SAS data library that contains only Version 8 sequential-format SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then selects the V8TAPE engine automatically. SAS also selects the V8TAPE engine automatically if you use the DCL DEFINE command to assign an OpenVMS logical name to an existing Version 8 sequential-format data library and then use that logical name as a libref in a SAS file specification.
- Set the value of the SEQENGINE= system option to V8TAPE. This option tells SAS which sequential engine to use as the default when no engine is specified and there are no existing SAS data sets.

Member Types Supported

The V8TAPE engine supports files with all Version 8 member types:

ACCESS
AUDIT
BACKUP
CATALOG
DATA
DMDB
FDB
INDEX
ITEMSTOR
MDDB
PROGRAM
PUTILITY
SASODS
UTILITY
VIEW

Using the V7 ENGINE

The section describes how to access Version 7 data libraries using the V7 engine. The V7 engine supports indexing and compression of observations.

When to Use the V7 Engine

Generally, SAS automatically determines the appropriate engine to use for accessing the files in the library. If you want to create a new library with an engine other than the default engine, you can override the automatic selection.

You use the V7 engine to create SAS data libraries on disk and to read from, write to, or update those libraries.

How to Select the V7 Engine

There are three ways to select the V7 engine:

- Specify V7 as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify V7TAPE in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.
- For an existing SAS data library on disk that contains only Version 7 SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then selects the V7 engine automatically. SAS also selects the V7 engine automatically if you use the DCL DEFINE command to assign an OpenVMS logical name to an existing Version 7 SAS data library on disk and then use that logical name as a libref in a SAS file specification.
- Set the value of the ENGINE= system option to V7. This option tells SAS which engine to use as the default when no engine is specified and there are no existing data sets, or when the directory is in mixed mode.

Member Types Supported

The V7 engine supports files with all Version 7 member types:

ACCESS
AUDIT
BACKUP
CATALOG
DATA
DMDB
FDB
INDEX
ITEMSTOR
MDDB
PROGRAM
PUTILITY
SASODS
UTILITY
VIEW

Engine/Host Options for the V7 Engine

The V7 engine provides several engine/host options that control the creating and access of SAS data sets. Most of the following options correspond to options that are available through OpenVMS Record Management Services (RMS).

You can use the following engine/host options with the V8 engine:

ALQ=
specifies the number of OpenVMS disk blocks to allocate initially to a data set when it is created. The value can range from 0 to 2,147,483,647. If the value is 0, the minimum number of blocks that is required for a sequential file is used. OpenVMS RMS always rounds the value up to the next disk cluster boundary.

The ALQ= option (allocation quantity) corresponds to the FAB$L_ALQ field in OpenVMS RMS. For additional details, see the data set option “ALQ=” on page 249 and *Guide to OpenVMS File Applications*.

ALQMULT=
specifies the number of pages that are preallocated to a file when it is created. The value can range from 1 to 128. The default value is 10. The ALQMULT= option is related to the ALQ= option.

For additional details, see the data set option "ALQMULT=" on page 250.

BUFSIZE=
specifies the size of the internal I/O buffer used for output data sets. The value can range from 0 to the maximum allowed under OpenVMS. The engine tries to use a value in the range of 8,192 to 32,768 if possible. The BUFSIZE= option must be specified in increments of 512.

For additional details, see the data set option "BUFSIZE=" on page 252.

CACHENUM=
specifies the number of I/O data caches used per SAS file. The value can range from 1 to 16. The default value is 10. The CACHENUM= option is used in conjunction with the CACHESIZ= option.

For additional details, see the data set option "CACHENUM=" on page 253.

CACHESIZ=
controls the size (in bytes) of the data cache used to buffer I/O pages. The value can range from 0 to 65,024. By default, the chosen value is an even multiple of the file's page size. A value of 0 specifies to use no data cache. Memory is consumed for the data cache, and multiple caches may be used for each data set opened. The disadvantage of large CACHESIZ= values is large consumption of memory. The advantage of large CACHESIZ= values is a reduction in the number of I/Os required to read from or write to a file.

The CACHESIZ= and BUFSIZE= options are similar, but they have important differences. The BUFSIZE= option specifies the file's page size, which is permanent. It can only be set on file creation. The CACHESIZ= option is the size of the internal memory cache used for the life of the current open, so it can change any time the file is opened. Also, BUFSIZE= cannot be used as an engine/host option; it is only valid as a data set option.

For additional details, see the data set option "CACHESIZ=" on page 253.

CNTLLEV=
specifies the level of shared access allowed to SAS data sets. Users can be given both read and write access to a data set that is opened for input only. By default, only shared read access is allowed.

For additional details, see the data set option "CNTLLEV=" on page 255.

DEQ=
specifies the number of OpenVMS disk blocks to add each time OpenVMS RMS automatically extends a data set during a write operation. The value can range from 0 to 65,535. OpenVMS RMS always rounds the value up to the next disk cluster boundary. A large value can result in fewer file extensions over the life of the file; a small value results in numerous file extensions over the life of the file. A file with numerous file extensions that may be noncontiguous slows record access.

If the value specified is 0, OpenVMS RMS uses the default value for the process.

The DEQ= option (default file extension quantity) corresponds to the FAB$W_DEQ field in OpenVMS RMS. For additional details, see the data set option "DEQ=" on page 256 and *Guide to OpenVMS File Applications*.

DEQMULT=
specifies the number of pages to extend a SAS file. The value can range from 1 to 128. The default value is 5.

For additional details, see the data set option "DEQMULT=" on page 257.

Using the V7TAPE Engine

In contrast to the V7 engine, the V7TAPE engine does not support indexing and compression of observations. Also, you cannot have a mixed mode, sequential-format SAS data library; you can store only Version 7, or Version 6 sequential SAS data sets on one tape, not some of each.

Note: For information about accessing SAS files on tape, see "Accessing SAS Files on Tape" on page 134. △

When to Use the V7TAPE Engine

Use the V7TAPE engine to create Version 7 sequential-format SAS data libraries either on disk or on tape and to access files in Version 7 sequential data libraries. The primary purpose of this engine is to enable you to back up Version 7 SAS data sets, catalogs, or whole data libraries. The V7TAPE engine makes it possible to back up applications that contain both SAS data sets and SAS catalogs.

In contrast to the V7 engine, the V7TAPE engine has the following limitations:

- Because the V7TAPE engine is a sequential engine, it cannot be used with the POINT= option of the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedure. If you want to use these features of the SAS language with sequential SAS data sets, then use the COPY procedure to copy them to a disk-format SAS data library. (See "Notes on Tape Usage" on page 136.)
- In a single DATA step or PROC step, you can use only one SAS data set from a particular sequential SAS data library.

How to Select the V7TAPE Engine

There are three ways to select the V7TAPE engine:

- Specify V7TAPE as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify V7TAPE in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.
- For an existing SAS data library that contains only Version 7 sequential-format SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then selects the V7TAPE engine automatically. SAS also selects the V7TAPE engine automatically if you use the DCL DEFINE command to assign an OpenVMS logical name to an existing Version 7 sequential-format data library and then use that logical name as a libref in a SAS file specification.
- Set the value of the SEQENGINE= system option to V7TAPE. This option tells SAS which sequential engine to use as the default when no engine is specified and there are no existing SAS data sets.

Member Types Supported

The V7TAPE engine supports files with all Version 7 member types:

ACCESS
AUDIT
BACKUP
CATALOG
DATA
DMDB
FDB
INDEX
ITEMSTOR
MDDB
PROGRAM
PUTILITY
SASODS
UTILITY
VIEW

Using the V6 Engine

This section describes how to access Version 6 data libraries using the V6 engine. These data libraries are disk-format data libraries, probably the most common type of libraries.

When to Use the V6 Engine

You use the V6 engine to create Version 6 SAS data libraries on disk and to read from, write to, or update those libraries. The V6 engine also enables you to index and compress observations. For more information about indexes and compressing observations, see *SAS Language Reference: Concepts*.

How to Select the V6 Engine

There are three ways to select the V6 engine:

- Specify V6 as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify V6 in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.
- For existing V6 SAS data libraries on disk that contain only V6 SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then examines the data set attributes and selects the V6 engine automatically.
- Set the value of the ENGINE= system option to V6. This system option tells SAS which engine to use as the default when either no engine is specified and there are no existing SAS data sets, or when the directory is in mixed mode.

Member Types Supported

The V6 engine supports files with all Version 6 member types:

ACCESS

CATALOG

DATA (with indexing capabilities)

DMDB

FDB

INDEX

MDDB

PROGRAM

VIEW

Engine/Host Options for the V6 Engine

The V6 engine provides several engine/host options that control the creation and access of SAS data sets. Most of the following options correspond to options that are available through OpenVMS Record Management Services (RMS).

You can use the following engine/host options with the V6 engine:

ALQ=
specifies the number of OpenVMS disk blocks to allocate initially to a data set when it is created. The value can range from 0 to 2,147,483,647. If the value is 0, the minimum number of blocks required for a sequential file is used. OpenVMS RMS always rounds the value up to the next disk cluster boundary.

The ALQ= option (allocation quantity) corresponds to the FAB$L_ALQ field in OpenVMS RMS. For additional details, see the data set option “ALQ=” on page 249 and *Guide to OpenVMS File Applications*.

CACHESIZ=
controls the size (in bytes) of the data cache used to buffer I/O pages. The value can range from 0 to 65,024. By default, the chosen value is an even multiple of the file's page size. A value of 0 specifies to use no data cache. Memory is consumed for the data cache, and multiple caches may be used for each data set opened. The disadvantage of large CACHESIZ= values is large consumption of memory. The advantage of large CACHESIZ= values is a reduction in the number of I/Os required to read from or write to a file.

The CACHESIZ= and BUFSIZE= options are similar, but they have important differences. The BUFSIZE= option specifies the file's page size, which is permanent. It can only be set on file creation. The CACHESIZ= option is the size of the internal memory cache used for the life of the current open, so it can change any time the file is opened. Also, BUFSIZE= cannot be used as an engine/host option; it is only valid as a data set option.

For additional details, see the data set option “CACHESIZ=” on page 253.

DEQ=
specifies the number of OpenVMS disk blocks to add each time OpenVMS RMS automatically extends a data set during a write operation. The value can range from 0 to 65,535. OpenVMS RMS always rounds the value up to the next disk cluster boundary. A large value can result in fewer file extensions over the life of

the file; a small value results in numerous file extensions over the life of the file. A file with numerous file extensions that may be noncontiguous slows record access.

If the value specified is 0, OpenVMS RMS uses the default value for the process.

The DEQ= option (default file extension quantity) corresponds to the FAB$W_DEQ field in OpenVMS RMS. For additional details, see the data set option "DEQ=" on page 256 and *Guide to OpenVMS File Applications*.

Data Set Options Supported by the V6 Engine

The V6 engine recognizes all the data set options that are documented in *SAS Language Reference: Dictionary* except the FILECLOSE= option. The engine/host options discussed in "Engine/Host Options for the V6 Engine" on page 150 can also be used as data set options when you use the V6 engine. For more information about data set options, see Chapter 12, "Data Set Options," on page 245.

System Option Values Used by the V6 Engine

The V6 engine uses the values of the BUFNO= and BUFSIZE= system options to determine permanent physical characteristics of output SAS data sets. For details about these system options, see "BUFNO=" on page 400 and "BUFSIZE=" on page 400.

Using the V6TAPE Engine

In contrast to the V6 engine, the V6TAPE engine does not support indexing and compression of observations. Also, you cannot have a mixed mode, sequential-format SAS data library; you can store only Version 6 or Version 5 sequential SAS data sets on one tape, not some of each.

Note: For information about accessing SAS files on tape, see "Accessing SAS Files on Tape" on page 134. △

When to Use the V6TAPE Engine

Use the V6TAPE engine to create sequential-format SAS data libraries either on disk or on tape and to access files in sequential data libraries. The primary purpose of this engine is to enable you to back up Version 6 SAS data sets, catalogs, or whole data libraries. The V6TAPE engine makes it possible to back up applications that contain both SAS data sets and SAS catalogs.

In contrast to the V6 engine, the V6TAPE engine has the following limitations:

- Because the V6TAPE engine is a sequential engine, it cannot be used with the POINT= option of the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedure. If you want to use these features of the SAS language with sequential SAS data sets, then use the COPY procedure to copy them to a disk-format SAS data library. (See "Notes on Tape Usage" on page 136.)
- In a single DATA step or PROC step, you can use only one SAS data set from a particular sequential SAS data library.

How to Select the V6TAPE Engine

There are three ways to select the V6TAPE engine:

- Specify V6TAPE as the value of the *engine* option in the LIBNAME statement or specify V6TAPE LIBNAME function, or in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.
- For an existing SAS data library that contains only Version 6 sequential-format SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then selects the V6TAPE engine automatically. SAS also selects the V6TAPE engine automatically if you use the DCL DEFINE command to assign an OpenVMS logical name to an existing Version 6 sequential-format data library and then use that logical name as a libref in a SAS file specification.
- Set the value of the SEQENGINE= system option to V6TAPE. This option tells SAS which sequential engine to use as the default when no engine is specified and there are no existing SAS data sets.

Member Types Supported

The V6TAPE engine supports files with all Version 6 member types:

ACCESS
CATALOG
DATA
DMDB
FDB
INDEX
MDDB
PROGRAM
VIEW

Engine/Host Options Used by the V6TAPE Engine

The V6TAPE engine recognizes all the data set options that are documented in *SAS Language Reference: Dictionary*. For more information about data set options, see Chapter 12, “Data Set Options,” on page 245.

Data Set Options Used by the V6TAPE Engine

The V6TAPE engine recognizes all the data set options that are documented in *SAS Language Reference: Dictionary* except for COMPRESS=, REUSE=, and CNTLLEV=. For more information about data set options, see Chapter 12, “Data Set Options,” on page 245.

System Options Used by the V6TAPE Engine

The V6TAPE engine uses the BUFNO= and BUFSIZE= system options. For information about these system options, see “BUFNO=” on page 400 and “BUFSIZE=” on page 400.

Using the V5 Engine

The V5 engine only provides read/input access to V5 files. You cannot create or update files.

When to Use the V5 Engine

You use the V5 engine to access SAS data sets that were created by Version 5 SAS software and that are not yet converted for use by a Version 6, Version 7, or Version 8 engine.

How to Select the V5 Engine

There are three ways to select the V5 engine:

- Specify V5 as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify V5 in the **Engine:** field of the New Library dialog box. You can open the New Library dialog box by issuing the LIBASSIGN command in the command window.
- For an existing SAS data library containing only Version 5 format SAS data sets, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. The SAS System then selects the V5 engine automatically.
- For new data libraries, set the value of the SYSTEM= system option to V5. This option sets the default engine to V5 when no engine is specified and the library is empty or the directory is in mixed mode.

Member Types Supported

The V5 engine supports files with the following member types:

CAT, GCAT, IMLWK, MODEL
: The V5 engine supports directory-level processing for these types of files (that is, the RENAME and DELETE functions in the DATASETS procedure and the DIR window). You must convert these files to Version 6 format before you can use them in Version 6 applications. Version 5 files of these types cannot be accessed in Version 7 or Version 8.

DATA
: The V5 engine fully supports SAS files of member type DATA.

Data Set Options Supported by the V5 Engine

The V5 engine supports all the data set options documented in *SAS Language Reference: Dictionary* except the COMPRESS=, REUSE=, and CNTLLEV= options. It also supports the engine-specific BUFSIZE= data set option. For details about the BUFSIZE= data set option, see “BUFSIZE=” on page 252.

System Options Supported by the V5 Engine

The V5 engine uses the BUFSIZE= system option, which is not a permanent attribute for Version 5 SAS data sets. The TAPECLOSE= system option is used for

Version 5 sequential-format SAS data libraries. For more information about these system options, see “BUFSIZE=” on page 400 and “TAPECLOSE=” on page 446.

System Utilities Used with the V5 Engine

You can copy Version 5 SAS data libraries using the COPY procedure running under Version 5. As long as the data library contains only SAS data sets, you can copy it using the COPY or DATASETS procedure running under Version 6.

The COPY or DATASETS procedure under Version 6 writes only SAS data files to a Version 5 data library. If your input library is a Version 6 data library containing a catalog, view, or access file, you receive a warning message indicating the file cannot be moved. However, the selected SAS data files are moved. If your input library is a Version 5 data library containing a member of type CAT, GCAT, IMLWK, or MODEL, you get a warning that the member cannot be moved, but all SAS data sets selected are moved.

Accessing Version 5 Sequential-Format Files on Disk

If you want to access Version 5 sequential-format files on disk, you must use the special libref TAPE*xxx* as you did in Version 5. You must use the V5 engine name in the LIBNAME statement when you assign this libref.

Using the CONCUR Engine

The concurrency (CONCUR) engine allows concurrent read and write access to data sets. Note that the concurrency engine supports only SAS data sets. It does not support SAS files of member types other than DATA, such as INDEX or CATALOG.

In contrast to the V8 engine, the CONCUR engine does not support indexing and compression of observations. The CONCUR engine can only access files within a single machine or OpenVMS cluster; access to SAS data sets on other operating environments and concurrent read/write access to SAS data sets across DECnet are features that are provided by SAS/SHARE software. For more information about using SAS/SHARE software, refer to *SAS/SHARE User's Guide*. The CONCUR engine is optimized for random concurrent access, while the V8 engine is better suited to sequential access. So, for example, if you intend to use the FSEDIT procedure or the POINT= option in the SET statement to access your data randomly, the CONCUR engine may be the best choice for you, even if you do not need any of the concurrent access capabilities.

Version 8 of the SAS System introduces support for several new features related to data sets. The CONCUR engine supports many of these features: member names with lengths up to 32 characters; variable names with lengths up to 32 characters; and member or variable labels with lengths up to 256 characters. Note that while the CONCUR engine supports the creation and access of Version 6 format files, the long character strings are not allowed when accessing or creating a Version 6 concurrency engine file. For more information about support for these longer character strings, see *SAS Language Reference: Concepts*.

How to Select the CONCUR Engine

There are three ways to select the CONCUR engine:

- Specify CONCUR as the value of the *engine* argument in the LIBNAME statement or LIBNAME function, or specify CONCUR in the **Engine:** field of the New Library dialog box.
- If you are sure your SAS data library contains only SAS data sets created with the CONCUR engine, do not specify a value for *engine* in the LIBNAME statement or LIBNAME function, or select **Default** as the type in the **Engine:** field of the New Library dialog box. SAS selects the CONCUR engine automatically.
- Set the value of the ENGINE= system option to CONCUR. This option indicates the default engine when either no engine is specified and there are no existing SAS data sets or the directory is in mixed mode.

The CONCUR engine creates and accesses SAS data sets in an acceptable format to allow record-locking and file-sharing.

CAUTION:
SAS data sets that are created with the CONCUR engine are not interchangeable with SAS data sets that are accessed and created with any other engine. If you plan to share a particular SAS data set, create it using the CONCUR engine. △

If you have a SAS data set that you want to share after it is created, you can copy it, using the CONCUR engine as the output engine. Then it will be in the correct format for sharing. For example, if you want shared update access to a data set that was created using the V8 engine, you can use the following statements to convert it:

```
libname inlib v8 '[mydir.base]';
libname outlib concur '[mydir.share]';
proc copy in=inlib out=outlib;
run;
```

After you run this SAS program, all SAS data sets that are created with the V8 engine in the data library that is referenced by INLIB are copied to the data library referenced by OUTLIB using the CONCUR engine. To create data sets using the CONCUR engine, your directory must have a version limit greater than 1.

Member Types Supported

The CONCUR engine supports the Version 8 member type DATA.

Engine/Host Options for the CONCUR Engine

Several concurrency engine options control the creation and access of SAS data sets. Most of these options have direct correlation to options available through OpenVMS Record Management Services (RMS). The CONCUR engine creates relative organization files with record-locking enabled.

Note: Data sets created with the CONCUR engine have a maximum observation length of 32K. △

You can use the following engine/host options with the CONCUR engine:

ALQ=
specifies the number of OpenVMS disk blocks to allocate initially to a data set when it is created. The value can range from 0 to 2,147,483,647. If the value is 0, the minimum number of blocks required for a sequential file is used. The ALQ=

option defaults to the bucket size. OpenVMS RMS always rounds the value up to the next disk cluster boundary.

The ALQ= option (allocation quantity) corresponds to the FAB$L_ALQ field in OpenVMS RMS. For additional details, see the data set option "ALQ=" on page 249 and *Guide to OpenVMS File Applications*.

BKS=

specifies the number of OpenVMS disk blocks in each bucket of the file. The value can range from 0 to 63. If the value is 0, the bucket size used is the minimum number of blocks needed to contain a single observation. The default value is 32.

When deciding on the bucket size to use, consider whether the file is usually accessed randomly (small bucket size), sequentially (large bucket size), or both (medium bucket size). The bucket size is a permanent attribute of the file, so this option applies to output files only.

The BKS= option (bucket size) corresponds to the FAB$B_BKS field in OpenVMS RMS or the FILE BUCKET_SIZE attribute when using File Definition Language (FDL). For additional details, see the data set option "BKS=" on page 251 and *Guide to OpenVMS File Applications*.

DEQ=

specifies the number of OpenVMS disk blocks to add each time OpenVMS RMS automatically extends a data set during a write operation. The value can range from 0 to 65,535. OpenVMS RMS always rounds the value up to the next disk cluster boundary. A large value can result in fewer file extensions over the life of the file; a small value results in numerous file extensions over the life of the file. A file with numerous file extensions that may be noncontiguous slows record access.

If the value specified is 0, OpenVMS RMS uses the default value for the process. The DEQ= option defaults to the bucket size.

The DEQ= option (default file extension quantity) corresponds to the FAB$W_DEQ field in OpenVMS RMS. For additional details, see the data set option "DEQ=" on page 256 and *Guide to OpenVMS File Applications*.

FILEFMT=

specifies the file format, or version of the engine, to use. Allowed values are 606, 607, 701, and 801. The default value is 801. There was an internal file format change between Release 6.06 and Release 6.07, and again between Version 6 and Version 7. The concurrency (CONCUR) engine can create and access all versions of the file format. When you access a file for input or update, the CONCUR engine detects the correct version of the existing file. When you create a new file, the CONCUR engine defaults to creating a Version 8 format file unless overridden by the FILEFMT= option.

The following example shows how to create a file in Release 6.07 format:

```
libname clib concur '[]';
data clib.v607 (filefmt=607);
. . . more SAS statements . . .
run;
```

HOSTFMT=

specifies the host platform format for a data set. The concurrency (CONCUR) engine can create and access files for both OpenVMS Alpha and OpenVMS VAX. Valid values are ALPHA or VAX, respectively. By default the data set is created in the native format of the platform on which SAS is running. You may use the HOSTFMT= option to specify that the data set should be created in a different representation. This is similar to using the Version 8 data set option OUTREP= to specify a data representation in a non-native format. The use of HOSTFMT= and

OUTREP= options is equivalent. HOSTFMT= is supported for compatibility with Version 6.

In the following example, the two data steps produce the same results:

```
data clib.vaxfile (hostfmt=vax);
. . . more SAS statements . . .
run;


data clib.vaxfile (outrep=vax_vms);
. . . more SAS statements . . .
run;
```

For more information about the OUTREP= data set option, see *SAS Language Reference: Dictionary*.

MBF=
specifies the number of I/O buffers you want OpenVMS RMS to allocate for a particular file. The value can range from 0 to 127, and it represents the number of buffers to use. By default, this option is set to 2 for files opened for update and 1 for files opened for input or output. If the value 0 is specified, the process' default value is used.

The MBF= option (multibuffer count) corresponds to the RAB$B_MBF field in OpenVMS RMS or the CONNECT MULTIBUFFER_COUNT attribute when using FDL. For additional details, see the data set option "MBF=" on page 260 and *Guide to OpenVMS File Applications*.

Data Set Options Supported by the CONCUR Engine

The CONCUR engine recognizes all data set options that are documented in *SAS Language Reference: Dictionary* except the FILECLOSE=, COMPRESS=, and REUSE options. Of special importance to the CONCUR engine is the portable data set option CNTLLEV=. (For details, see "CNTLLEV=" on page 255.) Other data set options that are likely to be useful include LOCKREAD= and LOCKWAIT=. (For details, see "LOCKREAD=" on page 258 and "LOCKWAIT=" on page 259.) For more information, refer to *SAS Language Reference: Dictionary*.

The engine/host options that are discussed in "Engine/Host Options for the CONCUR Engine" on page 155 can also be used as data set options when you use the CONCUR engine. For details, see "Specifying Data Set Options" on page 245.

System Option Values Used by the CONCUR Engine

The CONCUR engine does not use the values of any SAS system options.

DECnet Access

The CONCUR engine supports both creation and reading of files across DECnet, but not the updating of files across DECnet. You are allowed to create and read files because the engine uses multistreaming only when the file is opened for update. Support of DECnet access means you can now specify a node name in the physical pathname of your SAS data library, as long as you do not plan to update the data sets stored in the data library. The following is an example:

```
libname mylib concur 'mynode::bldgc:[testdata]';
```

Passwords

The CONCUR engine supports SAS System passwords. The syntax and behavior is the same as passwords used with the V8 (BASE) engine.

Internals of a Concurrency Engine Data Set

This section describes the internal structure of a concurrency engine data set. If you are familiar with OpenVMS RMS, it may be helpful to know the internal file format of a concurrency engine data set.

A concurrency engine data set is a relative format file. The record length is determined by the length of one observation, with a minimum length of 8 bytes. Because the data set is a relative format file, the maximum observation length of a concurrency engine data set is 32,767 bytes. The first portion of the file contains header records that provide information to the engine concerning the number of observations in the file, the number of variables, some positioning information to optimize access, the date and time, SAS System release, operating environment the data set was created on, and so forth.

Following the header information is information pertaining to each individual variable in the file. A NAMESTR is stored for each variable on the data set. The NAMESTR includes the variable name, type, label, and size. Multiple NAMESTRs are stored in a single record, up to the maximum number of NAMESTRs that the record length accommodates.

After the NAMESTRs, the observations begin. There is always one observation per record. With one exception, the record length is the observation length. If the observation length is less than 8 bytes, the record length defaults to 8. If you delete a record in a relative format file, the record still exists in the file, but it is marked as deleted.

Note: In a concurrency engine data set, a data set of deleted observations takes the same amount of disk space as a data set of valid observations. To remove the deleted observations, you must use the COPY procedure and copy the data set to a new data set type, such as a data set created with the V8 or V8TAPE engine. △

Although all record-locking capabilities are provided through the use of OpenVMS RMS features, some file-sharing capabilities are provided by OpenVMS RMS and some are provided by the engine itself. The engine can correctly set the share options of a file when the file is opened for input or update, because the SAS System uses the name of the existing data set directly. However, output data sets are created with a temporary name and then renamed to the actual data set name after the data set is closed. This ensures the integrity of existing data sets of the same name in case an error occurs during creation of the new data set. Therefore, the engine must handle all file-sharing issues that disallow sharing of output files. This is done through the locking of specific filenames, which is why your directory must have a version limit of at least 2 to create concurrency engine data sets.

Optimizing the Performance of the CONCUR Engine

Engine performance is often a trade-off between various factors. This section provides you with the necessary information so that you can optimize the performance of the CONCUR engine in your operating environment. By controlling the size and number of buffers, you can specify how the SAS System accesses your data. By specifying the data set options, you can control the level and amount of data that are accessed. The amount of disk space available for these operations also effects engine performance.

Controlling the Size and Number of Buffers

Depending on the type of record access your SAS application performs, you need to consider both the size of buffers (bucket size) and the number of buffers (multibuffer count). For complete details about specifying the size and number of buffers, see the BKS= and MBF= data set options in "BKS=" on page 251 and "MBF=" on page 260.

The two extremes of record access are records that are accessed completely sequentially or completely randomly. For example, many SAS procedures typically access data sets sequentially, processing the records from first to last. On the other hand, you may access observations in a completely random order when using the FSEDIT procedure to edit or browse observations in a data set.

There are also cases in which records are accessed randomly but may be reaccessed frequently. One example is an application that uses a data set in which particular observations contain information that is referred to frequently. Again, using the FSEDIT procedure as an example, the data set can be designed in such a way that you must access the first observation followed by observation 200, then the first observation again followed by observation 300, and so on.

Finally, there are cases in which records are accessed randomly, but then adjacent records are likely to be accessed. An application can use the POINT= option in a SET statement to selectively input the first 10 observations out of every 100 observations.

Most often, an application accesses a data set by a combination of several of these methods. The following list gives suggestions for the number of buffers and bucket size you should use for each method:

completely sequential or random access
: is most efficient with a single buffer. However, the bucket size differs:

  random access
  : is more efficient with a smaller bucket size.

  sequential access
  : is more efficient with a larger bucket size.

random access with reaccessed records
: is most efficient with multiple buffers to keep the reaccessed records in the buffer cache. You should use a small bucket size in this instance.

random access with subsequent adjacent access
: is most efficient with a single buffer. However, use a larger bucket size so that more records are stored in the buffer cache. This increases the probability that the required records have been read into memory with a single I/O.

If your program accesses the data set by several methods, you must find a compromise between the number of buffers and bucket sizes. This is what the SAS System attempts to do with the defaults, because the intended use of the file is unknown. Because you know the intended use of your CONCUR engine data sets, you can improve the CONCUR engine's performance by optimizing the buffer settings.

Using Portable Data Set Options

Several data set options are portable options that are available for all engines, but they are particularly useful in conjunction with the concurrency engine.

CNTLLEV=
: specifies the level of access (control level) to the data set, whether concurrent or exclusive. If you decide to create a concurrency engine data set to take advantage of its random access optimizations, but you do not need to provide for concurrent access at this time, you can use the CNTLLEV= data set option to further improve performance. By default, when using the concurrency engine, data sets that are

opened for input allow shared read access, data sets that are opened for output allow no sharing, and data sets that are opened for update allow shared update access. When sharing is allowed, record-level locking is enabled. When you do not need this feature, you can reduce the overhead of record locking by using CNTLLEV=MEM to disable the sharing.

The CNTLLEV= data set option takes one of two values:

MEM specifies that the application requires exclusive access to the data set. Member-level control restricts any other application from accessing the data set until the step has completed.

REC specifies that concurrent access is allowed and OpenVMS RMS record-level locking is enabled. This option entails more processing overhead and should be used only when necessary.

Each SAS procedure specifies a required control level to the engine, depending on the intended access of the observations. If you use CNTLLEV=REC and the SAS procedure requires member-level control to ensure the integrity of the data during processing, a warning is written to the SAS log indicating that inaccurate or unpredictable results can occur if the data set is being updated by another process during the analysis.

A common example of improving performance by overruling the CNTLLEV default of the procedure is with the FSEDIT procedure, which uses a default of CNTLLEV=REC. A session using the FSEDIT procedure with a concurrency engine data set does not need to incur the overhead of record-level locking if concurrent access is not required. By using the data set option CNTLLEV=MEM, the application tells the engine to override the control level specification of the procedure because exclusive access at the member level is desired. This disables record-level locking, decreases the overhead for processing the data set, and improves performance. In tests using the SET statement to input a concurrency engine data set, using the CNTLLEV=MEM option caused the step to run in one-third the CPU time as the same step using the CNTLLEV=REC option.

For syntax and usage examples for the CNTLLEV= data set option, see "CNTLLEV=" on page 255 and *SAS Language Reference: Dictionary*.

FIRSTOBS= and OBS=

specify a beginning and ending observation to subset your data set.

The value of the FIRSTOBS= data set option specifies the first observation that should be included for processing in the SAS DATA step. Some engines have to read the records sequentially, discarding them until the requested observation is reached. Because a concurrency engine data set is a relative format file, the engine can directly access the beginning observation without having to first read any other observations in the file.

Using the OBS= data set option to specify the last observation that you want to process can improve performance by terminating the input of observations without having to read records until the end-of-file character is reached.

For more information about the FIRSTOBS= and OBS= data set options, see *SAS Language Reference: Dictionary*.

Using the POINT= Option

You can use the POINT= option in a SET statement to access contiguous ranges of observation. For example, with the POINT= option, the SAS program can read observations 10 through 50, then observations 90 through 150, and so on. Obviously, only reading the records that you actually need improves performance by decreasing the number of records you must access. Due to the physical format of a concurrency engine data set, the engine can access the required records directly.

Disk Space Usage

For most data sets, the disk space that is required for a CONCUR engine data set and a V8 engine data set are comparable. However, for data sets in which the number of observations is greater than the number of variables, concurrency engine data sets are usually smaller. An exception to this is a concurrency engine data set that has many variables and only a few observations; in this case, space may be wasted.

However, there is a file format for both uncompressed and compressed data sets that makes the V8 engine disk space usage more efficient.

Performance Comparisons

Performance is a main concern for many applications, so it is useful to know how the CONCUR engine compares to the V8 engine when various features of the SAS System are used:

Creating data sets
: When you compare the creation and sequential input of data sets using each engine, the V8 engine tends to be faster when the data sets are small. However, as the size of the data set increases, the V8 and CONCUR engines are comparable in CPU time used. In all cases, the page faults that are incurred for the CONCUR engine are substantially less than for the V8 engine.

Accessing existing data sets
: When you compare random access of an existing file using both engines, the concurrency engine is much faster. When you use a large bucket size in the concurrency engine, with a comparable page size in the V8 engine, the concurrency engine takes approximately one-half as much CPU time. When the bucket size and page size are small, the concurrency engine takes about one-third as much CPU time. Again, page faults for the concurrency engine are substantially less.

Using the DBMS Interface Engines

To use the CA-OpenIngres, ORACLE, Oracle Rdb, or SYBASE engine, your site must license the corresponding SAS/ACCESS interface. In Version 8 of the SAS System, you specify these engines in the LIBNAME statement, the LIBNAME function, or the New Library dialog box, just as you do with the other engines. Each of these engines has DBMS-specific options that you can specify. For complete information about using these engines to access DBMS data, see *SAS/ACCESS Software for Relational Databases: Reference* and the following supplementary DBMS-specific chapters:

- *SAS/ACCESS Software for Relational Databases: Reference, (CA-OpenIngres Chapter)*
- *SAS/ACCESS Software for Relational Databases: Reference (ORACLE Chapter)*
- *SAS/ACCESS Software for Relational Databases: Reference (Oracle Rdb Chapter)*
- *SAS/ACCESS Software for Relational Databases: (SYBASE Chapter).*

Using the OSIRIS and SPSS Engines

The following read-only engines enable you to access files that were created with other vendors' software as if those files were written by the SAS System:

OSIRIS accesses OSIRIS files.

SPSS
: accesses SPSS files that were created under Release 9 of SPSS as well as SPSS-X system files and portable files.

You can use these engines in any SAS applications or procedures that do not require random access. For example, by using one of the engines with the CONTENTS procedure and its _ALL_ option, you can determine the contents of an entire SPSS file at once.

Restrictions on the Use of These Engines

Because these are sequential engines, they cannot be used with the POINT= option of the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedures in SAS/FSP software. However, you can use the COPY procedure, the DATASETS procedure, or a DATA step to copy an OSIRIS or SPSS file to a SAS data set, and then use either POINT= or SAS/FSP to browse or edit the file. Also, some procedures (such as the PRINT procedure) issue a warning message indicating that the engine is sequential.

Accessing OSIRIS Files

Although OSIRIS runs only under OS/390 (formerly MVS) and CMS, the SAS OSIRIS engine accepts an OS/390 data dictionary from any other operating environment that is running the SAS System. The layout of an OSIRIS data dictionary is the same on all operating environments. The data dictionary and data files should not be converted between EBCDIC and ASCII, however, because the OSIRIS engine expects EBCDIC data.

Assigning a Libref to an OSIRIS File

In order to access an OSIRIS file, you must use the LIBNAME statement or LIBNAME function to assign a libref to the file. (Alternately, you can select **Default** as the type in the **Engine:** field of the New Library dialog box.) Specify the OSIRIS engine in the LIBNAME statement as follows:

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

where

libref
: is a SAS libref.

OSIRIS
: is the OSIRIS engine.

data-filename
: is the physical name of the data file.

dictionary-filename
: is the physical filename of the dictionary file. The dictionary filename can also be a fileref or an OpenVMS logical name. However, if you use a fileref or an OpenVMS logical name for the *dictionary-filename*, do not use quotation marks.

You do not need to use a LIBNAME statement before running the CONVERT procedure if you are using PROC CONVERT to convert an OSIRIS file to a SAS data file. (For more information, see the procedure "CONVERT" on page 339.)

Note that the LIBNAME statement has no engine/host options for the SPSS engine.

If you previously assigned a fileref or an OpenVMS logical name to the OSIRIS file, then you can omit the *data-filename* in the LIBNAME statement. However, you must still use the DICT= option, because the engine requires both files. (For details, see "Example: Accessing OSIRIS Files" on page 163.)

You can use the same dictionary file with different data files. Enter a separate LIBNAME statement for each data file.

Referencing OSIRIS Files

OSIRIS data files do not have individual names. Therefore, for these files you can use a member name of your choice in SAS programs. You can also use the member name _FIRST_ for an OSIRIS file.

Under OSIRIS, the contents of the dictionary file determine the file layout of the data file. A data file has no other specific layout.

You can use a dictionary file with an OSIRIS data file only if the data file conforms to the format that the dictionary file describes. Generally, each data file should have its own DICT file.

Example: Accessing OSIRIS Files

Suppose you want to read the OSIRIS data file TEST1.DAT, and the dictionary file is TEST1.DIC. The following statements assign a libref to the data file and then run PROC CONTENTS and PROC PRINT on the file:

```
libname mylib osiris 'test1.dat' dict='test1.dic';
proc contents data=mylib._first_;
run;

proc print data=mylib._first_;
run;
```

Accessing SPSS Files

The SPSS engine supports portable file formats for both SPSS and SPSS-X files. The engine automatically determines which type of SPSS file it is reading and reads the file accordingly.

This engine can read only SPSS data files that were created under the same operating environment. For example, the SPSS engine under OpenVMS cannot read SPSS files that were created under the OS/390 operating environment. The only exception is an SPSS portable file, which can originate from any operating environment.

Assigning a Libref to an SPSS File

In order to access an SPSS file, you must use the LIBNAME statement or LIBNAME function to assign a libref to the file. (Alternately, you can select **Default** as the type in the **Engine:** field of the New Library dialog box.) Specify the SPSS engine in the LIBNAME statement as follows:

LIBNAME *libref* SPSS '*filename*';

where

libref
is a SAS libref.

SPSS
is the SPSS engine.

file-specification
is the physical filename.

You do not need to use a LIBNAME statement before running the CONVERT procedure if you are using PROC CONVERT to convert an SPSS file to a SAS data file. (For more information, see the procedure "CONVERT" on page 339.)

Note that the LIBNAME statement has no engine/host options for the SPSS engine.

If you previously assigned a fileref or an OpenVMS logical name to the SPSS file, then you can omit the *file-specification* in the LIBNAME statement. SAS uses the physical filename that is associated with the fileref or logical name. (For details, see "Example: Accessing SPSS Files" on page 164.)

Referencing SPSS Files

SPSS data files do not have names. For these files, use a member name of your choice in SAS programs.

SPSS data files have only one logical member per file. Therefore, you can use _FIRST_ in your SAS programs to refer to the first data file.

Example: Accessing SPSS Files

Suppose you want to read the SPSS file MYSPSSX.DAT. The following statements assign a libref to the file and then run PROC CONTENTS and PROC PRINT on the file:

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

proc print data=mylib._first_;
run;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OpenVMS Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. 518 pp.

SAS® Companion for the OpenVMS Environment, Version 8

1–58025–526–4

1st printing, October 1999

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

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