CHAPTER 2

Allocating SAS Data Libraries

Introduction

Under OS/390, a SAS data library can be a specially formatted operating environment data set that contains only SAS files. This data set may exist on disk, tape, or in a hiperspace.

Note: For the sake of clarity, please recall that a SAS data library is an operating environment data set and that a SAS file (also known as a SAS data set) is a member of a SAS data library. △

Only SAS can directly list, copy, rename, read, or modify the members of a SAS data library.
Chapter 2

For sites using OS/390 UNIX System Services in full-function mode, a SAS data library can also be a directory in an hierarchical file system (HFS). Any HFS directory can be used as a SAS data library, even if that directory contains files that are not managed by the SAS System. The SAS files in an HFS directory can be listed, copied, or renamed using operating environment facilities. However, due to their special structure, only SAS can properly read or modify the contents of a SAS file in an HFS directory. Filename extensions (the part of a filename that follows the final period) are specific to SAS and should not be modified.

Allocating an existing SAS data library simply makes it available to SAS and assigns it a logical name for the duration of your SAS session. Allocating a new SAS data library also creates the data library (except for external allocations of UNIX System Services directories). Therefore, in addition to assigning a logical name, you may also need to specify parameters such as block size, disposition, and space allocation. If you do not specify these additional parameters, SAS uses default values that are derived from SAS system options.

The logical name that you assign to your SAS data library is called a libref if you use the SAS LIBNAME statement, LIBNAME function, or SAS Explorer Library Assignment dialog box to allocate the library, or a DDname if you use the JCL DD statement or the TSO ALLOCATE command to allocate the library. Thereafter, you can use the libref or DDname as a convenient way of referring to the library.

Ways of Allocating SAS Data Libraries

Under OS/390, you can allocate a new or existing SAS data library in the following ways:

- internally (within a SAS session), using a LIBNAME statement, LIBNAME function, or SAS Explorer New Library Assignment dialog box.
- externally, using a JCL DD statement or a TSO ALLOCATE command.

See “Allocating SAS Data Libraries Internally” on page 32 and “Allocating SAS Data Libraries Externally” on page 35 for more information about these methods.

SAS Library Engines

The SAS System provides different engines that enable you to access and, in most cases, to update files of different types and different formats. See SAS Language Reference Dictionary for a complete discussion of the SAS System's multiple engine architecture and the different types of engines that are provided on all host operating environments.

When you allocate a SAS data library internally, you use the engine parameter of the LIBNAME statement or LIBNAME function to specify the appropriate engine for each SAS data library that you want to access. (See "LIBNAME Statement Syntax" on page 33.) Or SAS can determine which engine to use, based on the procedures described in "How SAS Assigns an Engine When No Engine Is Specified" on page 37.

When you allocate a SAS data library externally, SAS assigns an engine to the library the first time it is accessed by a SAS application. You can override the default by subsequently using the LIBNAME statement or LIBNAME function explicitly to associate a different engine with the library. See “Using the LIBNAME Statement or LIBNAME Function with Externally Allocated SAS Data Libraries” on page 37.

Table 2.1 on page 31 lists the library engines that have OS/390 and tells you where to look for more information about them.
### Table 2.1  SAS Library Engines for OS/390

<table>
<thead>
<tr>
<th>Engine</th>
<th>Description</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>V7</td>
<td>accesses SAS files in Version 7 disk format. You can also use the alias BASE to specify this engine.</td>
<td>“Using the V7 and V6 Engines” on page 51</td>
</tr>
<tr>
<td>V7TAPE</td>
<td>accesses SAS files in Version 7 sequential format, including sequential format on tape and sequential format on disk. You can also use the alias TAPE to specify this engine.</td>
<td>“Using the V7TAPE and V6TAPE Engines” on page 55</td>
</tr>
<tr>
<td>V6</td>
<td>accesses SAS files in Version 6 disk format</td>
<td>“Using the V7 and V6 Engines” on page 51</td>
</tr>
<tr>
<td>V6TAPE</td>
<td>accesses SAS files in Version 6 sequential format, including sequential format on tape and sequential format on disk.</td>
<td>“Using the V7TAPE and V6TAPE Engines” on page 55</td>
</tr>
<tr>
<td>V5</td>
<td>provides read-only access to SAS files in Version 5 disk format</td>
<td>Appendix 1 of SAS Companion for the MVS Environment, Version 6, Second Edition</td>
</tr>
<tr>
<td>V5TAPE</td>
<td>provides read-only access to SAS files in Version 5 sequential format, including sequential format on tape and sequential format on disk</td>
<td>Appendix 1 of SAS Companion for the MVS Environment, Version 6, Second Edition</td>
</tr>
<tr>
<td>BMDP</td>
<td>provide read-only access to BMDP, OSIRIS, and SPSS (including SPSS-X) files, respectively</td>
<td>“Introduction” on page 59</td>
</tr>
<tr>
<td>OSIRIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REMOTE</td>
<td>is used by SAS/CONNECT and SAS/SHARE software to access remote files</td>
<td>SAS/CONNECT User’s Guide and SAS/SHARE User’s Guide</td>
</tr>
<tr>
<td>XPORT</td>
<td>converts some SAS files to a format suitable for transporting from one operating environment to another</td>
<td>Moving and Accessing SAS Files across Operating Environments</td>
</tr>
</tbody>
</table>

### SAS View Engines

SAS view engines enable the SAS System to read SAS data views and DATA step views that are described by the DATA step, SQL procedure, or by SAS/ACCESS software. These engines support the SAS data set model only and are not specified in the LIBNAME statement or LIBNAME function:

- **ADB**
  accesses ADABAS database files.

- **DB2**
  accesses DB2 database files.

- **DDB**
  accesses CA-DATACOM/DB database files.

- **IDMS**
  accesses CA-IDMS database files.
IMS
  accesses IMS-DL/I database files.

DATASTEP
  accesses data sets that are described by a SAS DATA step.

These engines support the SAS data view and are also specified in the LIBNAME statement and the LIBNAME function:

ORACLE
  accesses ORACLE database files.

SQL
  accesses data sets that are described by the SQL procedure.

For more information about the SQL view engine, see SAS Guide to the SQL Procedure: Usage and Reference. For information about the other view engines, see the appropriate SAS/ACCESS software documentation.

Allocating SAS Data Libraries Internally

The LIBNAME statement or LIBNAME function allocates the operating environment data set, associates it with an engine, and assigns a libref to it. The assignment lasts for the duration of the SAS job or session unless you clear it. (See “Deallocating SAS Data Libraries” on page 42 for information about clearing a libref.)

Advantages of Allocating SAS Data Libraries Internally

Although you can use a JCL DD statement or a TSO ALLOCATE command to allocate SAS files externally, the LIBNAME statement or LIBNAME function can do much more. Here are several reasons why it is better to allocate SAS data libraries internally with the LIBNAME statement or function:

- The LIBNAME statement or function provides an easy way to do dynamic allocation in the batch environment. SAS programs that have LIBNAME statements or functions instead of external allocations can be executed either in the TSO environment or in the batch environment.
- In the batch environment, the LIBNAME statement or function provides the only means of creating a new SAS data library in the hierarchical file system of UNIX System Services.
- The JCL DD statement and the TSO ALLOCATE command are not portable to other operating environments. The LIBNAME statement or function is portable with minor changes to the physical-filename and options parameters.
- If you use the LIBNAME statement or function, you can allocate your data library for only as long as you need it, and then deassign and deallocate it. By contrast, DDnames that are allocated externally remain allocated for the duration of the SAS session or job. (The LIBNAME CLEAR statement deassigns an externally allocated libref, but it does not deallocate the file. See “Deallocating SAS Data Libraries” on page 42.)
- DDnames that are allocated externally cannot be reassigned later by a LIBNAME statement or LIBNAME function. You would receive an error message in the SAS log stating that the DDname is currently assigned.
By using macro statements and the LIBNAME statement or function, you can conditionally allocate files.

Using a LIBNAME statement or LIBNAME function allows you to specify an engine explicitly. Also, the following SAS engines must be specified in a LIBNAME statement or function because they are not assigned by default: XPORT, BMDP, SPSS, OSIRIS, and REMOTE.

DDnames that are allocated externally are not included in the list that is produced by the LIBNAME LIST statement nor in the SAS Explorer window until after they have been used as librefs in your SAS session. (See “Listing Your Current Librefs” on page 43.)

LIBNAME Statement Syntax

This section provides a brief overview of LIBNAME statement syntax. For complete information about the LIBNAME statement, see “LIBNAME” on page 315.

The general form of the LIBNAME statement is:

```
LIBNAME libref <engine> <'physical-filename'> < engine/host-options>;
LIBNAME libref <engine> <('physical-filename-1', ..., 'physical-filename-n')>
LIBNAME libref | _ALL_ CLEAR;
LIBNAME libref | _ALL_ LIST;
```

libref

is a logical name by which the library is referenced during your SAS session. The libref must begin with a letter and must contain 1-8 characters consisting of letters or numbers. SAS reserves some names as librefs for special system libraries. See “SAS System Files” on page 13 for more information.

When choosing a libref, follow the rules for SAS names, but do not use underscores. To read, update, or create files that belong to a permanent SAS data library, you must include the libref as the first part of a two-level SAS member name in your program statements, as follows:*  

```
libref.member
```

libref could also be a DDname that was specified in a JCL DD statement or in a TSO ALLOCATE command. The first time the DDname of a SAS data library is used in a SAS statement or procedure, SAS assigns it as a libref for the SAS data library.

engine

tells SAS which engine to use for accessing the library. Valid engine names for OS/390 include V7 (or its alias, BASE), V7TAPE, V6, V6TAPE, V5, V5TAPE, XPORT, REMOTE, BMDP, OSIRIS, and SPSS. See “SAS Library Engines” on page 30 for more information. If you do not specify an engine, SAS uses the procedures described in “How SAS Assigns an Engine When No Engine Is Specified” on page 37 to assign an engine for you. If the engine name that you supply does not match

* An exception is a SAS file in the USER library. In this case, you can use a one-level name. See “Directing Temporary SAS Data Sets to the USER Library” on page 15 for more information about the USER library.
the actual format or attributes of the data library, then any attempt to access the library will fail.

'physical-filename'

is the OS/390 operating environment data set name or the HFS directory name of the SAS data library, enclosed in quotes. See “Specifying Physical Files” on page 12. You can omit this argument if you are merely specifying the engine for a previously allocated DDname. Examples:

'userid.v7.library'
'MVS:userid.v7.library'
'HFS:/u/userid/v7/library'
'/u/userid/v7/library'

('physical-filename-1', ..., 'physical-filenamen')

is used to allocate an ordered concatenation of SAS data libraries and associate that concatenation with a single LIBREF. The concatenation can include direct-access bound libraries, UNIX System Services directories, and sequential libraries, as long as all of the libraries in the concatenation can be accessed with the specified engine.

When accessing a member of a concatenated series of libraries, SAS searches through the concatenation in the order that it was specified. SAS accesses the first member that matches the specified name. SAS will not access any members with the same name that are positioned after the first occurrence in the concatenation.

'engine/host options'

are options that apply to the SAS data library. The host-specific options that are available depend on which engine you are using to access the data library. See “SAS Library Engines” on page 30 for more information about SAS engines. Specify as many options as you need. Separate them with a blank space. For a complete list of available options, see “LIBNAME” on page 315.

LIBNAME Statement Examples

Allocating an existing SAS data library:

The following LIBNAME statement allocates the existing SAS data library LIBRARY.CATALOG.DATA and assigns the libref BOOKS to it:

libname books 'library.catalog.data';

The following LIBNAME statement allocates the existing SAS data library contained in the UNIX System Services directory /corp/dev/test1:

libname results '/corp/dev/test1';

The following LIBNAME statement allocates a concatenation of SAS data libraries:

libname concatlb ('library.catalog.data', '/corp/dev/test1');

Allocating a new SAS data library:

The following LIBNAME statement allocates the new SAS data library prefix.NEW.SASDATA. The prefix will be taken from the value of the SYSPREF =
Allocating SAS Data Libraries

system option, as explained in “Specifying Physical Files” on page 12. The LIBNAME statement assigns the libref NEWLIB to the data library.

```
libname newlib '.new.sasdata'
disp=(new,catlg)
  unit=3380 vol=xyz828 space=(cyl,(10,3))
  blksize=23040;
```

Because the operating environment data set did not previously exist, appropriate values are specified for DISP=, UNIT=, and other engine/host options. The engine name is not specified explicitly, so SAS assigns the default engine to the libref. (The default engine is the engine that is specified by the SAS system option ENGINE=.) SAS uses these values to dynamically allocate the data set; then it assigns the libref to the data set.

**Note:** If you do not specify default values for DCB attributes when you allocate a new operating environment data set with the LIBNAME statement, the SAS System supplies default values. See “Internal Allocation: V7 Engine” on page 53 and “Internal Allocation: V7TAPE Engine” on page 57 for details.

Specifying additional options for a previously allocated SAS data set:

See “Using the LIBNAME Statement or LIBNAME Function with Externally Allocated SAS Data Libraries” on page 37.

---

Allocating SAS Data Libraries Externally

There are several advantages to allocating SAS data libraries internally (see “Advantages of Allocating SAS Data Libraries Internally” on page 32). However, you can also use either a JCL DD statement or a TSO command to allocate a SAS data library externally and to assign a DDname to it.

**Note:** If you do not use the LIBNAME statement or LIBNAME function to specify an engine for a data set that was allocated externally (as described in “Using the LIBNAME Statement or LIBNAME Function with Externally Allocated SAS Data Libraries” on page 37), then SAS uses the procedures described in “How SAS Assigns an Engine When No Engine Is Specified” on page 37 to determine which engine to use.

---

JCL DD Statement Examples

Allocating an existing SAS data library:

The following JCL DD statement allocates the cataloged data set LIBRARY.CATALOG.DATA and assigns the DDname BOOKS to it:

```
//BOOKS DD DSN=LIBRARY.CATALOG.DATA,
//   DISP=OLD
```

The following JCL DD statement allocates an existing SAS data library, which is stored in an UNIX System Services directory:

```
//HFSLIB DD PATH=’/corp/dev/test1’
```

Note that UNIX System Services recognizes and distinguishes between uppercase and lowercase letters in pathnames.
Allocating a new SAS data library:

This example allocates a new SAS data library on tape:

```
//INTAPE DD DSN=USERID.V7.SEQDATA,
// UNIT=TAPE,LABEL=(1,SL),
// DCB=(RECFM=U,LRECL=32756,BLKSIZE=32760),
// DISP=(NEW,KEEP),VOL=SER=XXXXXX
```

Notice that DCB attributes are specified. When you allocate a new SAS data library externally, you must either specify DCB attributes or accept the default DCB attributes that SAS supplies. See “DCB Attributes for Direct Access Bound Libraries with the V7 Engine” on page 52 and “DCB Attributes for Direct Access Bound Libraries with the V7TAPE Engine” on page 56 for details.

Specifying additional options for a previously allocated SAS data library:

See “Using the LIBNAME Statement or LIBNAME Function with Externally Allocated SAS Data Libraries” on page 37.

---

**TSO ALLOCATE Command Examples**

Allocating an existing SAS data library:

The following TSO ALLOCATE command allocates the cataloged data set LIBRARY.CATALOG.DATA and assigns the DDname BOOKS to it:

```
alloc file(books) da('lib.cat.data') old;
```

The following command performs the same allocation, this time using the SAS X statement to submit the TSO ALLOC command (see “X” on page 323 for details about the X statement):

```
x alloc file(books) da('lib.cat.data') old
```

The following commands create a new UNIX System services directory and allocate that new directory as a SAS data library with the DDname RESULT2:

```
x mkdir '/corp/dev/test2');
x alloc file(result2) path('/corp/dev/test2');
```

Allocating a new SAS data library:

The following example allocates a new sequential SAS data library on disk:

```
alloc fi(intape) da(v7.seqdata) dsorg(ps)
  recfm(u) blksize(6144) new
```

Notice that DCB attributes are specified. When you allocate a new SAS data library externally, you must either specify DCB attributes or accept the default DCB attributes that SAS supplies. See “DCB Attributes for Direct Access Bound Libraries with the V7 Engine” on page 52 and “DCB Attributes for Direct Access Bound Libraries with the V7TAPE Engine” on page 56 for details.

Specifying additional options for a previously allocated SAS data library:

See “Using the LIBNAME Statement or LIBNAME Function with Externally Allocated SAS Data Libraries” on page 37.
Using a DDname as a Libref

After a DDname has been assigned, you can use it in a SAS job in the same way you would use a libref. For example:

```sas
proc contents data=books._all_; run;
```

The first time the DDname BOOKS is used in this manner, SAS assigns it as a libref for the SAS data library.

When a DDname is allocated externally, it is not listed by the LIBNAME LIST statement or in the SAS Explorer until after you have used it as a libref in your SAS session. (See “Listing Your Current Librefs” on page 43.)

Using the LIBNAME Statement or LIBNAME Function with Externally Allocated SAS Data Libraries

You can use the LIBNAME statement or LIBNAME function to specify an engine for a data library that was previously allocated externally. For example, suppose you used an X statement to submit the following TSO ALLOCATE command, which allocates the SAS data library QUARTER1.MAILING.LIST:

```tsocmd
x alloc f(mail) da('quarter1.mailing.list') old;
```

You could later use the LIBNAME statement to associate an engine with the library as follows:

```sas
libname mail v7seq;
```

This LIBNAME statement or LIBNAME function associates the Version 7 sequential engine with the data library that is referred to by the DDname MAIL. You can then read to and write from the sequential format data library QUARTER1.MAILING.LIST. You do not need to specify the data set name in this example, as long as MAIL is the DDname for QUARTER1.MAILING.LIST. If you do specify the data set name, the name must match the data set name that was already allocated to that DDname.

How SAS Assigns an Engine When No Engine Is Specified

In some cases, you may choose not to specify an engine name in the LIBNAME statement or LIBNAME function, or you may choose not to issue a LIBNAME statement or function for a data library that was allocated externally. For these situations, you need to know how SAS assigns an engine to the library.

- For a new SAS data library on disk, SAS uses the value of the ENGINE= system option as the engine. (Exception: For a new sequential-format data library on disk, you must include the engine name in the LIBNAME statement or LIBNAME function.)
- For an existing SAS data library on disk, SAS examines the library's DCB attributes to determine which engine to use.
- If DSORG=PS and RECFM=FS, SAS opens the library and inspects the header information to determine if it should use the V6 engine or V7 engine. Because this operation can degrade performance, it is more efficient, if possible, to specify the engine in the LIBNAME statement or LIBNAME function.
Allocating OS/390 Generation Data Sets

If DSORG=DA, SAS uses the V5 engine.

If DSORG=PS and RECFM=U, SAS uses the current value of the SEQENGINE= system option.

If either DSORG= or RECFM= is undefined, SAS uses the current value of the ENGINE= system option.

For new tape data sets, SAS uses the current value of the SEQENGINE= system option.

For existing tape data sets, instead of looking at the characteristics of the tape, SAS opens the file and reads the first record to determine whether it should use V7TAPE, V6TAPE, or V5TAPE.

For an existing directory in UNIX System Services, SAS uses the V7 engine automatically.

Allocating OS/390 Generation Data Sets

An OS/390 generation data set (or generation) is a version of a data set that is stored as a member of a generation data group. (For detailed information about generation data groups, see your IBM documentation.) Both SAS data libraries and standard external files can be stored and managed as generation data groups. The methods of allocating and accessing generations are the same for SAS data libraries as they are for external files. For more information, see “Allocating Generation Data Sets” on page 70 and “Allocating a Multivolume Generation Data Group” on page 42.

Note: Do not confuse OS/390 generation data sets with SAS generation data sets, a new feature in Version 7. For more information on SAS generation data sets, see SAS Language Reference: Concepts.

Allocating Multivolume SAS Data Libraries

A non-HFS Version 7 SAS data library on disk may span more than one volume. The operating environment data set that contains the SAS data library may exist on multiple DASD volumes, but it is processed by SAS software as one logical entity. This capability greatly increases the storage capacity of a data library.

There is no SAS limit on the number of DASD volumes that a data library may occupy. You will likely take one of two general approaches to allocating new multivolume libraries, depending on why you need them. If you have an extremely large data library, requiring more than one DASD volume, you will probably want to preallocate the library on the desired volumes, most likely in full volume increments. If you have medium to large libraries requiring less than a full volume of space, and you need multivolume support because your available DASD space is fragmented across many volumes, then you will probably want to allocate them dynamically. Both allocation methods are described below. Support for the dynamic allocation of multivolume data libraries is new in Version 7.

General Guidelines

When creating a multivolume SAS data library, observe the following guidelines:
A data library can span as many volumes as the operating environment will allow. See your IBM documentation for details.

When you first allocate a data library, you are not required to specify all of the volumes that the library will ever occupy. You can add more volumes as a data library grows.

A multivolume data library must be allocated on devices of the same type (for example, all 3380s or all 3390s). However, the control units that are attached to the devices may support a mix of standard CKD and extended CKD (ECKD) capabilities. SAS software optimizes I/O operations at the device level by exploiting the capabilities of the primary control unit that is attached to the volume that is being accessed.

To maintain the integrity of your multivolume libraries, it is extremely important to ensure that they are backed up, migrated, and restored properly. The most failsafe method of backing up and restoring a multivolume library is to use the COPY procedure for both the backup and restore operations, specifying DISP=OLD on the multivolume library allocation. You can also use IEBGENER or an equivalent utility with an exclusive allocation of the multivolume library.

If your installation runs periodic full-volume backups, you must ensure that the other volumes in a multivolume data library are not updated while a backup of one of the volumes is running; otherwise, data integrity cannot be guaranteed if you attempt to restore the library from the full-volume backups. The best way to ensure data integrity is to back up the data library as a complete unit in volume sequence order and to restrict data library access to “read only” during the backup operation. This ensures integrity if a restore operation is required, because the data library can be restored logically in volume sequence order.

When restoring a multivolume data library, you do not need to restore the library to the same volumes on which it previously existed. A data library can even be restored to multiple volumes of a different device type than it previously existed on, as long as the restore utility places the maximum number of blocks of the given block size on each track of the new device.

Preallocating New Multivolume Libraries

These examples show some typical multivolume preallocations. They all show full-volume space allocations for a 3390, because this would be a typical case for preallocating large data libraries, but any allocation size will work. Note that multivolume libraries are not required to consume whole volumes. Also, because SAS will attempt to acquire secondary space allocations only on the last volume of a multivolume data library, you should not specify secondary space allocations on preallocated libraries. (IEFBR14 is an IBM utility program that simply returns immediately, allowing you to force the system through normal batch step allocation/deallocation processing.)

The following JCL will preallocate a three-volume 3390 data library:

```
//ALLOC EXEC PGM=IEFBR14
//VOL1 DD DSN=MY.PAYROLL.LIBRARY,DISP=(NEW,KEEP),
  // DCB=(DSORG=PS,RECFM=FS,LRECL=27648,
  // BLKSIZE=27648),UNIT=3390,
  // SPACE=(CYL,1113),VOL=SER=PR0001
//VOL2 DD DSN=MY.PAYROLL.LIBRARY,DISP=(NEW,KEEP),
  // DCB=(DSORG=PS,RECFM=FS,LRECL=27648,
  // BLKSIZE=27648),UNIT=3390,
```
Dynamically Allocating New Multivolume Data Libraries

These examples show some typical dynamic multivolume allocations. The resulting data libraries will contain only as many volumes as the application requires. Only the primary space allocation amount will be obtained at allocation time. SAS will acquire the secondary allocations as they are required, just as with a single-volume data library, but additional volumes will also be acquired whenever secondary allocations are no longer possible on the (current) last volume.

The following JCL allocates a large temporary WORK library, using up to three volumes, if required. Note that there must be three available units in the system, even if they do not all get space on them, for this example to work, because the system chooses the candidate volumes at allocation time.
Allocating SAS Data Libraries

Allocating an Existing Multivolume SAS Data Library

After a multivolume SAS data library has been initially allocated or extended to additional DASD volumes, any of the following statements can be used for allocation as long as the data library is cataloged in volume sequence order.

```
libname usevols 'sas.mvdatlib';
//USEVOLS DD DSN=SAS.MVDATLIB,DISP=OLD
x alloc fi(usevols) da('sas.mvdatlib') shr;
```

Cataloging a multivolume SAS data library ensures that SAS can process it properly without requiring the VOL=SER= list each time the library is allocated for use.

The information in this section applies to the SMS environment as well. There are no special requirements for allocating existing multivolume data libraries in the SMS environment.
Allocating a Multivolume Generation Data Group

As explained in “Allocating Generation Data Sets” on page 70, a SAS data library can be stored and managed as a generation data group (GDG). To allocate a GDG that spans multiple volumes, use either of the following methods:

**SMS**
Define the GDG with a standard IDCAMS job. Ensure that the job does not contain a DD card for the data set; otherwise, the DEFINE GENERATION DATA GROUP statement generates duplicate data set messages and fails. (If you use the GUARANTEED SPACE attribute in an SMS environment, data set information is allocated on all designated packs; the DEFINE recognizes that information has been duplicated and fails.) When creating a new generation, use the relative form of the data set name in your DD card.

**IEFBR14**
Define the GDG with a standard IDCAMS job. When creating a new generation, observe the guidelines provided in “Allocating Multivolume SAS Data Libraries” on page 38 and use the absolute form of the data set name (for example, DSN=XXX.MULTI.GDG.G0001V00).

Assigning Multiple Librefs to a Single SAS Data Library

You can assign more than one libref to the same SAS data library. Any assigned libref may be used to access the data library. In fact, you can use the librefs interchangeably.

For example, suppose that in two different programs you used different librefs for the same data sets. Later you develop a new program from parts of the two old programs, or you include two different programs with the %INCLUDE statement. In the new program, you could simply assign the two original librefs to each data set and proceed.

Note: Even when multiple librefs are assigned to the same SAS data library, the SAS data library is allocated only once. The first logical name (DDname) that you assign is used as the DDname in the operating environment allocation.

Deallocateing SAS Data Libraries

The method you use to deallocate a SAS data library depends on how it was allocated.

- To deallocate a SAS data library that was allocated with a LIBNAME statement or LIBNAME function, issue a LIBNAME statement or function in the following form, using the libref for the data library that you want to deallocate:
  
  ```
  LIBNAME libref <CLEAR>;
  ```

  This statement deassigns the libref and deallocates the library only if no other librefs are assigned to that library.

  Note: Librefs that were assigned by a LIBNAME statement or LIBNAME function are cleared automatically at the end of your SAS session.

- To deallocate a library that was allocated with a TSO command, first issue a LIBNAME statement or LIBNAME function to “clear” (deassign) the libref, as
shown above. Then issue an operating environment command that “frees” (deallocates) the data set.

For example, suppose that a SAS data library with the libref MYLIB is stored in the operating environment data set MYID.RECENT.DATA. The following two statements would clear the libref and deallocate the operating environment data set:

```bash
libname mylib clear;
x free da('myid.recent.data');
```

If the data library is currently being used by a DATA step or PROC, the attempt to deallocate the library fails.

You can deallocate a SAS data library in the SAS EXPLORER window by selecting the DELETE menu.

Note: If multiple librefs have been assigned to the same operating environment data set, the data set is not deallocated until all librefs assigned to the data set have been cleared.

---

**Listing Your Current Librefs**

You can use either the LIBNAME command or a form of the LIBNAME statement or LIBNAME function to list your currently assigned librefs.

- When you issue the LIBNAME command, the SAS Explorer window is displayed. The SAS Explorer window lists all the librefs that are currently assigned for your session.

  The SAS Explorer window displays the full operating environment data set name of the SAS data library, as well as the engine that is used to access the data library.

- The following form of the LIBNAME statement writes to the SAS log the attributes of all the librefs that are currently assigned for your session:

  ```bash
  LIBNAME _ALL_ LIST;
  ```

---

**Estimating the Size of a SAS Data Set**

Under OS/390, a non-HFS SAS data library is a single file that contains the library's SAS data sets. You cannot obtain information from the file system on how large each member SAS data set is.

To obtain a rough estimate of how much space you need for a disk-format, non-HFS SAS data set that was created by the V7 engine, follow these steps:

Note: This procedure is valid only for uncompressed native SAS library members that were created with the V7 engine.

1. Using the CONTENTS procedure, specify the DIRECTORY option and using the DATA option to specify the SAS library member.

2. Using the output of the CONTENTS procedure, divide the Data Set Page Size statistic for the SAS library member by the Blocksize statistic for the directory to obtain the number of blocks per page.

3. Multiply the number of blocks per page by the Number of Data Set Pages statistic for the SAS library member to obtain the number of blocks.
4 Increase the number of blocks by 5% to account for overhead associated with the library directory control structures associated with the SAS library member.

5 Divide the increased number of blocks by the Blocks Per Track statistic for the directory, then round up to the nearest whole track.

Note:

- The number of tracks required for a member of a SAS data library can vary depending on the track size of the device on which the data library resides, on the block size of the SAS data library's operating environment data set, and on the data set page size for the member.

- The Max Obs per Page statistic for the member may be used to adjust the Number of Data Set Pages statistic in step 3 above to estimate the space required to add a number of new observations. Divide the number of new observations by the Max Obs per Page statistic to get the number of new data set pages needed. Then add the number of new pages to the Number of Data Set Pages statistic.

To determine the rough size in bytes of the allocation needed for a SAS library member, follow these steps:

1 Run the CONTENTS procedure using the DATA option to specify the SAS library member.

2 Using the output of the CONTENTS procedure, multiply the Pagesize value for the SAS library member by the number of pages for the member to obtain a number of bytes.

3 Increase the number of bytes by 5% to account for overhead associated with the library directory control structures associated with the library member.

For additional information about estimating the size of a SAS data set, see Tuning SAS Applications in the MVS Environment, by Michael Raithel. This book is available from SAS Institute as part of the Books by Users program.

---

Using the CONTENTS Procedure to Determine Observation Length

To determine the length of each observation in a Version 7 SAS data set, you can create a Version 7 SAS data set that contains one observation. Then run the CONTENTS procedure to determine the observation length. The CONTENTS procedure displays engine/host-dependent information, including page size and the number of observations per page for uncompressed SAS data sets. For example, the following input produces a SAS data set plus PROC CONTENTS output:

data oranges;
input variety $ flavor texture looks;
cards;
navel 9 8 6 ;
proc contents data=oranges;
run;

The output is shown in Output 2.1 on page 45.
Allocating SAS Data Libraries

Estimating the Size of a SAS Index

Output 2.1 CONTENTS Procedure Output

The only values that you need to pay attention to are **Observation Length** and **Compressed**:

**Observation Length**
- is the record size in bytes.

**Compressed**
- has the value NO if records are not compressed; it has the value YES if records are compressed. (If the records are compressed, do not use the procedure given in “Estimating the Size of a SAS Data Set” on page 43.)

### Estimating the Size of a SAS Index

Under OS/390, you can use the following formula to obtain a rough estimate of the size of a simple index:

\[
\text{size} = (\text{unique}) \times (8 + \text{length}) \times (1.15)
\]

**unique**
- is the number of unique values of the key variable.

**8**
- is the number of bytes of overhead for each unique value in the index.
length

is the length of the key variable, in bytes.

1.15

is a multiplication factor for general index overhead.

Note: The result of this calculation will be in bytes. You can convert this value to tracks based on the capacity of your data storage device.