CHAPTER

2

Moving SAS Files

Choosing a Method for Moving a SAS File

Consider these factors when choosing a method to move a SAS file between hosts:

- the architectural compatibility of the hosts
- the release of SAS software that is running on each host
- the SAS member to be moved between the hosts
- whether you have a license for SAS/CONNECT.

Host Architectural Compatibility

The method you choose for moving a file depends on the host compatibility. *Compatible* hosts share the same internal architectures; they store numeric or

character data in the same way. *Incompatible* hosts do not share a common internal architecture; each stores numeric or character data differently from the other.

Here are examples of compatible and incompatible host types:

Compatible Hosts	Incompatible Hosts
OS/390 and CMS	OS/390 and Windows
32-bit UNIX RISC hosts[1]	64-bit RISC host COMPAQ Digital UNIX and 32-bit UNIX RISC hosts[1]
Version 6 Windows and Version 6 OS/2	Version 7 Windows and Version 8 OS/2[2]

[1] HP-UX, Solaris, AIX, and MIPS ABI are examples of 32-bit UNIX RISC hosts.

[2] Version 7 and Version 8 file formats are identical.

For a complete list of compatible host groups that share an identical architecture, store numeric data identically, or store character data identically, see Chapter 16, "Architectural Compatibility," on page 119.

Host Type and SAS Release

The type of host and the SAS release that it runs determine which SAS files can be moved. For example, you can move a SAS file from a CMS host that runs Release 6.07 to a Windows NT host that runs Version 8. See "Host Types Supported According to SAS Release" on page 4 for a list of supported hosts and SAS releases.

Note: The ability to regress a SAS file (for example, from Version 8 to Version 6) also depends on the member being moved and the particular method used. For details about moving a SAS file from a later release to an earlier release, see "Regressing SAS Data Sets from Version 8 to Version 6 Format" on page 19. △

SAS Members That Can Be Moved

You can move these SAS members between hosts:

- Data set
- Catalog (and most catalog entries)
- PROC SQL view
- DMDB (Data mining database)[1]
- MDDB (Multi-dimensional database)[1]
- FDB (Financial database)[1]

[1] Version 8 traditional methods to move SAS files do not support these members. However, SAS/CONNECT does support them. For details about using SAS/CONNECT to move files between hosts, see Chapter 6, "Using SAS/CONNECT to Transport Files between Hosts," on page 45.

SAS Members That Cannot Be Moved

Here are examples of SAS files that traditional transport methods cannot move between hosts:

- Some catalog entries
- DATA step view

- SAS/ACCESS view
- Item stores

The only certain method for finding out if you can move a member type is to try it. The SAS log reports whether the move operation for the member type is not supported. Here is a typical error message for an attempted move of an unsupported catalog entry type:

```
WARNING: Entry type KEYMAP is not supported by CPORT.
```

The preceding message means that PROC CPORT could not move the catalog entry type KEYMAP. Because PROC CPORT is the only supported traditional technique for moving catalogs, you can infer that this catalog entry type cannot be moved. For details about supported traditional techniques for moving files, see "Traditional Move Operations" on page 12.

As another example, the move technique that you attempt may not support a given member type. Here is a typical error message for an attempted move with an unsupported move technique:

```
NOTE: Copying SOURCE.TESTCAT to XPTOUT.TESTCAT
  (memtype=CATALOG).
ERROR: The CATALOG FILE OPEN function is not supported
  by the XPORT engine.
ERROR: File XPTOUT.TESTCAT.CATALOG has not been saved
  because copy could not be completed
```

The preceding message means that the XPORT engine with PROC COPY cannot be used for moving a catalog. If one method fails, you may try another one. For information about supported methods, see "Traditional Move Operations" on page 12.

Neither a DATA step view nor a SAS/ACCESS view can be moved. However, you can re-create a view after its source data has been transferred to and restored at the target host. For example, a DATA step view must be re-created at the target host after the data set is restored at the target host.

Note: A PROC SQL view can be created without the underlying table in place. However, if you invoke PROC SQL and submit a query that uses the view, an error message is displayed. △

Or, you can move the original SAS statements that were used to define your IML module definitions to the target host and then re-create the modules there. However, you cannot retrieve the definitions from your original IML module. For an item store, you must re-create it on the target host.

Third-Party DBMS Files That Cannot Be Moved

The traditional transport methods do not support moving third-party DBMS files between hosts, which include relational databases (DB2 and ORACLE are examples), nonrelational databases (ADABASE is an example), and PC file formats Excel and Lotus.

However, SAS/CONNECT and SAS/SHARE support accessing certain third-party DBMS files by means of the SQL procedure Pass-Through facility, which requires a SAS/ACCESS software license.

For more information about SAS/CONNECT, see "Is SAS/CONNECT Licensed?" on page 12. For more information about SAS/SHARE, see Chapter 7, "Using SAS/SHARE to Access Remote Files," on page 55.

For complete details about SAS/ACCESS, see the document that is appropriate for the particular DBMS that you use. For information about relational databases, see

SAS/ACCESS Software for Relational Databases: Reference. For details about nonrelational databases, see *SAS/ACCESS Interface to CA-DATACOM/DB: Reference*, *SAS/ACCESS Interface to IMS-DL/I Software*, *SAS/ACCESS Interface to CA-IDMS Software: Reference*, or *SAS/ACCESS Interface to SYSTEM 2000 Data Management Software: Reference*, as appropriate.

Is SAS/CONNECT Licensed?

SAS/CONNECT, which requires a separate license, is an alternative solution for file transport. It offers an effective method for moving SAS files and external files (binary or text) between incompatible hosts that run any SAS release.

Note: SAS/CONNECT also supports moving certain third-party DBMS files by means of the SQL procedure Pass-Through facility, which requires a SAS/ACCESS software license. △

Using the UPLOAD and DOWNLOAD procedures, you move SAS files between hosts for remote processing. For incompatible hosts, SAS/CONNECT performs a dynamic file format translation between the native formats of the hosts for each read, write, or update operation. Likewise, SAS/CONNECT transparently converts the format that is associated with a SAS release that runs on one host to the format that is associated with the release that runs on the other host, depending on whether PROC UPLOAD or PROC DOWNLOAD is used. The SAS/CONNECT built-in transport facility bypasses the explicit creation of a transport file, which must be manually transferred to the target host and restored there. For more information about SAS/CONNECT, see Chapter 6, “Using SAS/CONNECT to Transport Files between Hosts,” on page 45.

Traditional Move Operations

Here are the traditional types of move operations:

- Transport
- Copy
- Convert

Transporting a File between Hosts

The process of moving a file between incompatible hosts is known as *transporting*; for example, from CMS to Windows. Such a process is necessary in order to account for binary incompatibilities between different host architectures.

Because compatible hosts share a common internal architecture, you *do not* need to transport SAS files between them. Instead, you just copy the file from one host to the other.

CAUTION:
Do not unnecessarily transport a file between hosts. The possibility for loss of numeric precision increases each time a file is translated to or from transport format. Therefore, perform the transport operation only when moving a file between incompatible hosts. △

The process of moving a file between incompatible hosts, where the SAS releases are different, is known as *converting*; for example, between CMS running at Release 6.12 and Windows running at Version 8. The file is automatically converted to the more

recent release as it is transported. In this instance, you do not perform an explicit conversion operation.

Note: The direction of a SAS file release conversion (from an earlier to a later release or from a later to an earlier release) is restricted by the particular transport method used. For more information, see "Traditional Methods for Creating and Restoring Files in Transport Format" on page 15. △

Copying a File between Hosts

In order to move a file between *compatible* hosts (such as HP-UX and Solaris) that run the same or different SAS release, you use the appropriate communications software commands to copy the file over the network to the target host. No transporting or converting is performed. In order to access the file, at the target host, you use the LIBNAME statement to assign the location (such as the directory) of the transferred file.

Deciding Whether to Convert the Copied File

If the SAS release that the target host runs (for example, Version 8) and the format of the file that was copied to the target host (for example, Version 6) are different, then you must decide whether you want to convert the copied file to the release that is running on the target host.

You must convert a Version 6 file to Version 8 format under these circumstances:

- To use Version 8 data set features (such as long variable names, integrity constraints, or data set generations). For complete information about Version 8 features, see *SAS Language Reference: Concepts*.
- To update catalogs.
- To access DATA step views.

However, if you do not plan to use Version 8 features, you can still read, write, and update Version 6 files from a Version 8 session.

An advantage of not converting is that you do not have to copy the file to a Version 8 library. If only Version 6 files reside in a library, you can access the files by using the LIBNAME statement. SAS automatically invokes the appropriate engine to allow file access.

A disadvantage of accessing Version 6 files from a Version 8 session is that mixed libraries can result. Although Version 6 files and Version 8 files can exist in the same library for most hosts, their management can be difficult.

Note: The OS/390 host does not support mixing Version 6 and Version 8 files in the same library. △

For example, global operations that you intentionally apply to an entire library are, in fact, restricted to either the Version 6 files or the Version 8 files in the library, but not to both.

Note: To find out the contents of a library, use PROC DATASETS. For details about PROC DATASETS, see *SAS Procedures Guide*. To see the filename extensions of files in a directory which forms a library, use the file list command for your operating system. △

In order to create Version 8, Version 7, or Version 6 files, you must specify the appropriate engine option in the LIBNAME statement. To subsequently access the file, you may optionally specify the appropriate engine option. Here are examples:

```
LIBNAME grades v6 'SAS-data-library';
LIBNAME grades v8 'SAS-data-library';
```

```
LIBNAME grades 'SAS-data-library;
```

On supported hosts, the specified engine locates the appropriate files according to the filename extension. For example, a Version 6 engine on a Windows host locates Version 6 data sets whose filename extension is **.sd2**. A Version 8 engine on a Windows host locates Version 8 data sets whose filename extension is **.sas7bdat**.

Note: Version 7 and Version 8 filename extensions are identical. △

Note: CMS and OS/390 hosts do not support filename extensions. △

If you omit an engine option from a statement that is executed in a Version 8 SAS session, SAS automatically invokes the engine that is appropriate to the files in the library. For example, if all files are appended with a Version 8 filename extension, the V8 engine is invoked. Likewise, if only files that have a Version 6 filename extension are present, the V6 engine is invoked. However, if a mix of files inhabit the library, then the native engine for the SAS session being run is used. For example, a Version 8 session invokes the V8 engine to access only Version 8 files, ignoring Version 6 files.

For a list of Version 8 and Version 6 filename extensions, see Chapter 17, "SAS Filename Extensions and File Headers," on page 125.

Converting a Version 6 File to Version 8 Format

Use the COPY procedure to convert a Version 6 file to a Version 8 format. This action converts the SAS file to the newest release without transporting its internal representation. Here is a UNIX example:

```
libname shipped v6 '/dept/612lib';
libname new v8 '/dept/701_lib';
proc copy in=shipped out=new;
run;
```

The LIBNAME statements include the appropriate engines that define the format of the files in either Version 6 or Version 8 format. PROC COPY copies the SAS files in the Version library that is identified in the IN= option to a Version 8 library that is identified in the OUT= option.

Accessing a Version 6 File in a Version 8 Mixed Library

If you do not want to convert a Version 6 file to Version 8 format, you have no more steps to perform. To access the Version 6 file on a Version 8 host, you specify the LIBNAME statement with the V6 engine. Here is a UNIX example:

```
libname shipped V6 '/dept/801_lib';
```

Note: If you access Version 6 files in a Version 8 environment, you cannot use features that are specific to Version 8. △

Note: Omission of the V6 engine in this LIBNAME statement would cause the V8 engine, which is the base engine, to be used. The V8 engine would locate only Version 8 files. △

Converting a File on the Same Host

In order to convert (or upgrade) a file from an earlier release to a later release on a single host, then you explicitly convert the file. For example, you use the COPY procedure to upgrade your SAS system files from Release 6.12 to Version 8 format.

Instructions for performing this operation are not given here because this book addresses operations between hosts that may be optionally attached to a network. This conversion operation is confined to a single host. See *SAS Language Reference: Concepts* for complete information.

Summary of Traditional Move Operations

Table 2.1 on page 15 gives a summary of the traditional move scenarios and associated operations to be performed.

Table 2.1 Actions Required for Traditional SAS File Move Operations

Between Compatible Hosts?	Between Different SAS Releases?	Example	Operation Performed
no	yes	UNIX 6.12 to CMS 8	Transport with Implicit Convert
no	no	UNIX 8 to CMS 8	Transport
yes	yes	HP-UX 6.12 to AIX 8	Convert[1]
yes	no	HP-UX 7 to AIX 8[2]	Copy
yes	yes	HP-UX 6.12 to HP-UX 8 (same host)	Convert[3]

[1] If you do *not* convert Version 6 files to Version 8 format, you may access Version 6 files in a Version 6 or Version 8 environment. However, you are limited to using features of Version 6.

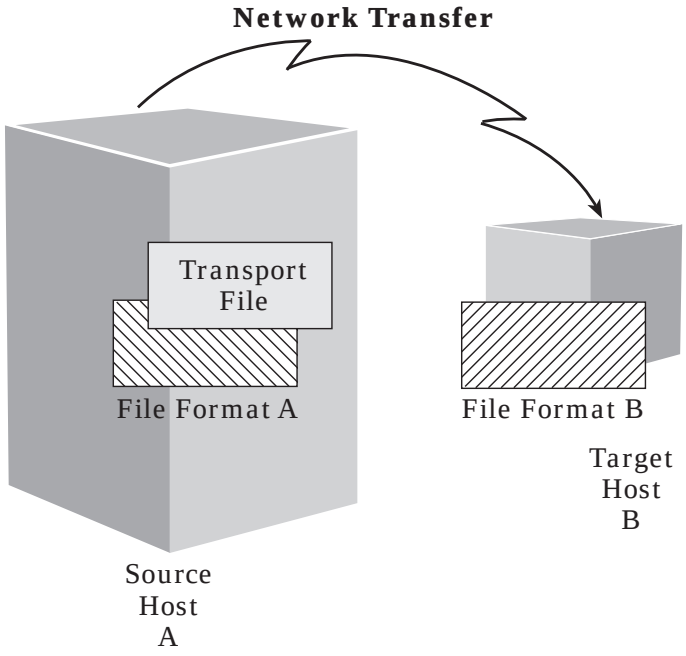
[2] Version 7 and Version 8 file formats are identical.

[3] See *SAS Language Reference: Concepts* for details about this operation. This book addresses all other cases.

Traditional Methods for Creating and Restoring Files in Transport Format

Figure 2.1 on page 16 depicts how traditional methods move files between hosts:

Figure 2.1 Creating and Restoring a Transport File



This figure shows the creation of a transport file as a means to reconcile incompatible host file formats. The transport file is transferred across the network and the target host restores the file to its native format.

Note: In addition to transferring a transport file over the network, you can also transfer a file to a magnetic medium that can be physically mounted on the target host. △

You can use the following SAS statements and procedures to create and restore a transport file, as appropriate.

XPORT engine with DATA step or PROC COPY
: At the source host, the LIBNAME statement with the XPORT engine and either the DATA step or PROC COPY creates a data file in transport format. At the target host, the same method is used to translate the transport file into the target host native format.

CPORT and CIMPORT procedures
: At the source host, PROC CPORT writes data sets or catalogs or both data sets and catalogs to transport format. At the target host, PROC CIMPORT translates the transport file into the target host native format.

Table 2.2 on page 17 summarizes the methods, according to member type, that can be used to create transport files.

Table 2.2 Supported SAS Member Types and Traditional Transport Methods

SAS Member Type	XPORT Engine with either DATA step or PROC COPY	PROC CPORT and PROC CIMPORT
Data set	•	•
Catalog		•

Benefits of Using the XPORT Engine with PROC COPY

The benefit of using the XPORT engine (with either the DATA step or the COPY procedure) is the ability to move files between hosts, regardless of whether you are moving the transport file to a later or an earlier SAS release.

Note: Regressing a data set (moving from a later release to an earlier release) destroys the features that are specific to the later release. For example, when moving from Version 8 to Version 6, Version 8 long variable names are truncated to eight bytes. For details about file regression, see "Regressing SAS Data Sets from Version 8 to Version 6 Format" on page 19. △

Using the XPORT engine is preferable when sending a transport file to a destination host whose SAS release is unknown.

Another advantage of using the XPORT engine is that you can create the transport file once and direct it to multiple target hosts that run different releases.

The primary reason for using the XPORT engine with the DATA step is to dynamically create one or more data sets, to order them, and then to translate them to transport format. By contrast, PROC COPY allows you to translate multiple data sets that already exist in a library.

Limitations of Using the XPORT Engine with PROC COPY

The limitation of the XPORT engine is that it supports only members of type DATA. It does not support members of type CATALOG.

Furthermore, the XPORT engine supports a Version 5-compatible feature set. For example, the XPORT engine cannot support Version 8 features such as long variable names. Warning or error messages report limitations that are encountered during the transport operation. For information about typical error messages and recovery actions for this problem, see "File *library.member*.DATA has too long a member name for the XPORT engine" on page 109.

The XPORT engine with PROC COPY does not support the transport of any type of view, MDDBs, or DMDBs.

Benefit of Using PROC CPORT and PROC CIMPORT

The CPORT and CIMPORT procedures are preferable for moving members of both type DATA and CATALOG.

Limitations of Using PROC CPORT and PROC CIMPORT

The disadvantage of using PROC CPORT and PROC CIMPORT is that they do not allow the file transport operation from a later version to an earlier version, which is

known as regressing (for example, from Version 8 to Version 6). PROC CPORT and PROC CIMPORT move files only from an earlier version to a later version (for example, from Version 6 to Version 8) or between the same versions (for example, from one Version 8 host to another Version 8 host).

However, you can move files between releases of Version 6; for example, from Release 6.12 to Release 6.08. For details about using PROC CPORT and PROC CIMPORT to move files between Version 6 releases, see the Version 6 SAS Language Reference.

PROC CPORT and PROC CIMPORT do not support the transport of any type of view, MDDBs, or DMDBs.

Alternatives to Traditional Methods

None of the traditional transport methods supports any type of view (such as the PROC SQL view), MDDBs, or DMDBs. However, SAS/CONNECT does support these additional member types. For details about SAS/CONNECT, see Chapter 6, "Using SAS/CONNECT to Transport Files between Hosts," on page 45.

For file access only, besides supporting members of type DATA and CATALOG, SAS/SHARE also supports PROC SQL views and MDDBs. For details about SAS/SHARE, see Chapter 7, "Using SAS/SHARE to Access Remote Files," on page 55.

For file access only, besides supporting members of type DATA and CATALOG, CEDA also supports MDDBs for read-only access. For details about CEDA, see Chapter 8, "Using Version 8 Cross-Environment Data Access (CEDA)," on page 65.

Interpreting a Filename Extension

A filename extension reflects these data file traits:

- Engine (V6, V7, or V8) that was used to create the file

  Note: The V7 and V8 engines behave identically. △
- Host architecture on which the file was created
- Member type of the file.

Note: All Version 8 hosts, except CMS and OS/390, use filename extensions. See "Host Types Supported According to SAS Release" on page 4 for a complete list of Version 8 hosts. For CMS and OS/390, you can use PROC CONTENTS to find out the file's member type and the engine that was used to create it. △

Here are examples of filename extensions:

SAS Engine	UNIX (HP-UX)	OpenVMS	Windows
	Member of Type DATA		
V6	.ssd01	.SASEB$DATA	.sd2
V7 or V8	.sas7bdat	.sas7bdat	.sas7bdat

For a complete list of SAS Version 6, Version 7, and Version 8 filename extensions according to host type, see Chapter 17, "SAS Filename Extensions and File Headers," on page 125.

Note: Version 7 and Version 8 filename extensions are identical. △

Using a Non-Native Engine to Create Files of Type DATA

SAS provides the flexibility to create a data file in a format other than the default format of the current SAS session. For example, from a Version 8 session, you can invoke the V6 engine to create a data file in Version 6 format. You can create a file in a format that is only at the same release or at an earlier release than that of the current SAS session. You cannot invoke the V8 engine from a Version 6 SAS session.

Note: The V8 engine and the V7 engine are identical. △

This flexibility accommodates concurrent data access by users on hosts that run SAS under different releases.

Regressing SAS Data Sets from Version 8 to Version 6 Format

The only traditional method for regressing a data set from Version 8 to Version 6 is by using PROC COPY. However, if SAS/CONNECT is licensed, you can use PROC UPLOAD and PROC DOWNLOAD to regress a data set.

Version 8 support of long variable names, long variable labels, and long data set labels may make Version 8 data sets incompatible with Version 6 data sets. In order to revert back to Version 6, these long names must be truncated to a length that is supported in Version 6. Here are the truncation rules:

Version 8 Data Set Object Names to Regress	Truncates to *x* characters for Version 6
Data set labels	40
Variable labels	40
Variable names	8

In order to transport Version 8 files back to Version 6, set the portable VALIDVARNAME system option to the value V6 in the SAS session in which you are transporting the file. Here are examples, which are specified in the form of a SAS system option and a macro variable:

```
options VALIDVARNAME=V6
%let VALIDVARNAME=V6;
```

For details about setting the VALIDVARNAME system option, see *SAS Language Reference: Dictionary*.

The truncation algorithm that is used to produce the 8-character variable name also resolves conflicting names:

- The first name that is greater than 8 characters is truncated to 8 characters. A truncation from PROPERTYTAXRATE to PROPERTY is the first truncation.
- The next name that is greater than 8 characters is truncated to 8 characters. If it conflicts with an existing variable name, it is truncated to 7 characters, and a suffix of 2 is added. For example, PROPERTYTAXRATE is truncated to PROPERT2.
- The suffix is increased by 1 for each truncated name that conflicts with an existing name. If the suffix reaches 9, the next conflicting variable name is truncated to 6 characters, and a suffix of 10 is appended. For example, PROPERTYTAXRATE is truncated to PROPER10.

The VALIDVARNAME option solves the long variable name truncation problem. However, there are no techniques for regressing these Version 8 features to Version 6:

- Data set names that exceed 8 characters
- Integrity constraints
- Data set generations.

The solution to regressing data sets with these features is to re-create the data sets without the Version 8 features in a Version 8 session.

Note: No traditional methods support regressing catalogs from Version 8 to Version 6. However, SAS/CONNECT does support moving *some* catalog entries from Version 8 to Version 6. See Chapter 6, "Using SAS/CONNECT to Transport Files between Hosts," on page 45 for details. △

Transport Format

Using a supported method, you create a transport file that is represented internally in a common language format, which is known as *transport format*. A transport file is a sequential file that contains one or more data sets or catalogs or both in transport format. A transport file contains a header (which describes the content of the file) and the contents of the member types (which are represented in binary format).

Two distinctive types of transport formats result from the particular method that creates the transport file:

- XPORT engine
- CPORT and CIMPORT procedures.

Because these methods produce transport files whose formats are different, you cannot mix methods to create and then to restore the transport file. The methods that you use must be identical or be a compatible pair. For example, you can create and restore a transport file using the XPORT engine and PROC COPY or the DATA step at both the source and target hosts. Likewise, you can create a transport file using PROC CPORT at the source host, and you can import the transport file using PROC CIMPORT at the target host. However, you *cannot*, for example, create a transport file using the XPORT engine and PROC COPY at the source host and then try to use PROC CIMPORT to import the transport file at the target host.

Determining Which Method Was Used to Create the Transport File

How you determine the traditional method (XPORT engine with PROC COPY or PROC CIMPORT and PROC CIMPORT) that was used to create a transport file depends on your host.

- Hosts that store character data in ASCII format

  Use a text editor or an operating system read or view command to read the file.

  The XPORT engine creates a file whose first 40 characters contain this ASCII text:

  ```
  HEADER RECORD*******LIBRARY HEADER RECORD!!!!!!!00
  ```

  PROC CPORT creates a file whose first 40 characters contain this ASCII text:

  ```
  **COMPRESSED** **COMPRESSED** **COMPRESSED** **COM
  ```

  Note: If you set the NOCOMPRESS option to PROC CPORT, compression is suppressed, which prevents the display of the preceding text in a transport file. △

  For technical details about the transport format that is used for a data set, see Technical Support article TS-140, *The Record Layout of a SAS Transport Data Set*.

- Hosts that store character data in EBCDIC format

  Transport files that are created on hosts that represent character data in EBCDIC (OS/390 and CMS) are not readable. For details about how to interpret ASCII data on EBCDIC hosts, see “Interpreting EBCDIC as ASCII Data” on page 122.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Moving and Accessing SAS Files across Operating Environments, Version 8*, Cary, NC: SAS Institute Inc., 1999. 186 pages.

Moving and Accessing SAS Files across Operating Environments, Version 8

ISBN 1-58025-480-2

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

1st printing, October 1999