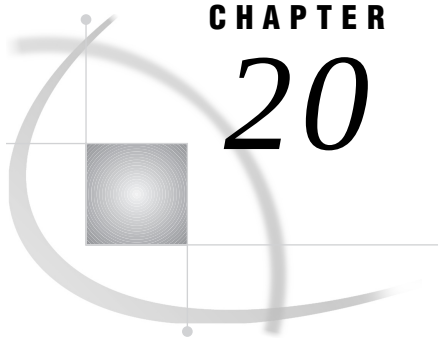

CHAPTER

20

Examples

The Examples

This chapter gives detailed examples that show how to create, transfer, and restore transport files between two hosts. Table 20.1 on page 134 describes the basic characteristics of each example:

Table 20.1 Summary of the Examples

Members to Move	From Source Host and SAS Release	To Target Host and SAS Version	Using SAS Procedure
Data sets	OpenVMS Alpha 6.12	HP-UX 7	XPORT engine with PROC COPY
Data sets and catalogs	OS/390 6.09	Windows NT 7	PROC CPORT and PROC CIMPORT
Data sets	OS/390 TSO 7	UNIX 7	PROC CPORT and PROC CIMPORT
Data sets	JCL Batch OS/390 6.09	UNIX 7	XPORT engine with PROC COPY

Although the examples are host-specific, the fundamental SAS command syntax for all transport methods is identical across host types. Furthermore, although the examples support a Version 7 environment, they also apply to Version 8. The noteworthy syntax difference among host types is how you specify the SAS data library name in the LIBNAME statement. For complete details about the syntax for the LIBNAME statement, see your operating environment companion.

OpenVMS Alpha to HP-UX File Transport

Using PROC COPY at the Source Host to Create Transport Files

The following example shows a SAS program that creates three data sets in OpenVMS Alpha format and translates them to transport format.

Example Code 20.1 SAS Program That Creates Data Sets and Transport Files

```
❶ libname xptlib xport 'xptlib.dat';
❷ libname xptds xport 'xptds.dat';

   /* creates data set GRADES; contains numeric and */
   /* character data */
❸ data grades;
     input student $ test1 test2 final;
  datalines;
  Fred 66 80 70
  Wilma 97 91 98
  ;

  /* creates data set SIMPLE; contains */
  /* character data only */
❹ data simple;
     x='dog';
     y='cat';
     z='fish';
  run;
```

```
  /* creates data set NUMBERS; contains */
  /* numeric data only */
❺ data numbers;
      do i=1 to 10;
         output;
      end;
   run;

 /* create a transport file for the entire library */
❻ proc copy in=work out=xptlib;
  run;

 /* create a tranport file for a dataset */
❼ proc copy in=work out=xptds;
     select grades;
  run;
```

1 The LIBNAME statement assigns the libref XPTLIB to the physical location XPTLIB.DAT, which stores the entire library to be created. The XPORT engine creates XPTLIB.DAT.

2 The LIBNAME statement assigns the libref XPTDS to physical location XPTDS.DAT, which stores the single data set to be created. The XPORT engine creates XPTDS.DAT.

3 The DATA step creates the data set WORK.GRADES, which contains two observations. Each observation contains four variables (one character and three numeric values).

4 The DATA step creates a second data set WORK.SIMPLE, which contains a single observation. The observation contains three character values.

5 The DATA step creates a third data set WORK.NUMBERS, which contains ten observations. Each observation contains a single numeric value.

6 PROC COPY copies all three data sets from the default WORK library to the new library XPTLIB. The WORK data sets are written to the output library XPTLIB in transport format.

7 PROC COPY copies the selected data set GRADES to the new library XPTDS. The data set GRADES is written to output library XPTDS in transport format.

Viewing the SAS Log at the Source Host

The following example shows a SAS log that documents the successful execution of the SAS program in “Using PROC CPORT at the Source Host to Create Transport Files: OS/390” on page 146.

Example Code 20.2 Source Host SAS Log

```
❶ NOTE: SAS (r) Proprietary Software Release 6.12  TS050

❷ NOTE: Running on DEC Model 7000 MODEL 740 Serial Number 80000000.

❸ NOTE: Libref XPTLIB was successfully assigned as follows:
      Engine:        XPORT
      Physical Name: Device:system-specific file/pathname XPTLIB.DAT
❹ NOTE: Libref XPTDS was successfully assigned as follows:
      Engine:        XPORT
      Physical Name:system-specific file/pathname XPTDS.DAT
```

```
❺ NOTE: The data set WORK.GRADES has 2 observations and 4 variables.
NOTE: The data set WORK.SIMPLE has 1 observations and 3 variables.
NOTE: The data set WORK.NUMBERS has 10 observations and 1 variables.
❻ NOTE: Copying WORK.GRADES to XPTLIB.GRADES (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across dissimilar engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPTLIB.GRADES has 2 observations and 4 variables.
NOTE: Copying WORK.NUMBERS to XPTLIB.NUMBERS (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across dissimilar engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPTLIB.NUMBERS has 10 observations and 1 variables.
NOTE: Copying WORK.SIMPLE to XPTLIB.SIMPLE (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across dissimilar engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPTLIB.SIMPLE has 1 observations and 3 variables.
❼ NOTE: Copying WORK.GRADES to XPTDS.GRADES (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across dissimilar engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPTDS.GRADES has 2 observations and 4 variables.
```

1 The source host runs SAS Release 6.12, which means that the SAS session default library engine is V612.

2 The source host is a DEC Model 7000, which refers to the AX7000 (OpenVMS Alpha).

3 SAS assigns the libref XPTLIB to the physical device whose specification is platform-dependent. The XPORT engine creates XPTLIB.

4 SAS assigns the libref XPTDS to the physical device whose specification is platform-dependent. The XPORT engine creates XPTDS.

5 The first three notes in this series report the creation of the data sets WORK.GRADES, WORK.SIMPLE, and WORK.NUMBERS.

6 The next series of notes report that SAS copies WORK.GRADES to XPTLIB.GRADES, WORK.NUMBERS to XPTLIB.NUMBERS, and WORK.SIMPLE to XPTLIB.SIMPLE. The XPORT engine translates each data set from OpenVMS Alpha format to transport format.

Note: The following notes about the SAS system option BUFSIZE do not indicate an error condition. BUFSIZE specifies the permanent buffer size for an output data set, which can be adjusted to improve system performance. The system value that is assigned to the BUFSIZE option is used because the XPORT engine does not support the BUFSIZE= option. See your operating environment companion documentation for details. △

7 SAS copies WORK.GRADES to XPTDS.GRADES. The XPORT engine translates the data set from OpenVMS Alpha format to transport format.

Verifying Transport Files

You are advised to verify the integrity of your transport files at the source host before the files are transferred to the target host. A successful verification at the source host can eliminate the possibility that the transport file was created incorrectly. Also, after you transfer the transport file to the target host, you can compare the transport file that was sent from the source host with the file that was received at the target host. See “Methods for Verifying Transport Files” on page 160 for details.

Transferring the Transport Files to the Target Host

Before you transfer a transport file to the target host, verify its file attributes. The following example shows typical output:

Example Code 20.3 Using DIR/FULL to Verify the Attributes of the Transport File

```
vms> DIR/FULL xptlib.dat
Directory HOSTVAX:[JOE.XPTTEST]

   XPTLIB.DAT;1                  File ID:  (31223,952,0)
   Size:            7/8          Owner:    [HOSTVAX,JOE]
   Created:   30-SEP-1999 16:47:31.34
   Revised:   30-SEP-1999 16:47:31.69 (1)
   Expires:      Effective:    File organization:  Sequential
   Shelved state:      Online
   File attributes:    Allocation: 8, Extend: 0, Global buffer count: 0
                        Version limit: 2
   ❶ Record format:      Fixed length 80 byte records
   ❷ Record attributes:  None
   RMS attributes:     None
   Journaling enabled: None
   File protection:    System:RWED, Owner:RWED, Group:RE, World:
   Access Cntrl List:  None

   Total of 1 file, 7/8 blocks.
   $ dir/size xptlib.dat

   Directory HOSTVAX:[JOE.XPTTEST]

   XPTLIB.DAT;1               7

   Total of 1 file, 7 blocks.
```

1 The OpenVMS VAX RECORD FORMAT attribute indicates a fixed record type and an 80-byte record size. These values are required for a successful file transfer across the network.

An OpenVMS Alpha host RECORD FORMAT should indicate a record length of 512 bytes.

2 The RECORD ATTRIBUTES field should contain the value NONE.

CAUTION:

If this field contains CARRIAGE RETURN CARRIAGE CONTROL, file corruption results. To prevent corruption before you transfer the transport file, remove this value from the RECORD ATTRIBUTES field. An error message alerts you to this condition after you attempt to transfer the corrupted file. △

After you verify the attributes of a transport file, use FTP to transfer the transport file to the target host.

In this example, the target host retrieves the transport file from the source host because the source host does not have permission to write to the target host directory. A source host is unlikely to have permission to write a transport file to a target host.

At the target host, change the directory to the location where the transport file will be copied. The following example shows how to use FTP commands to get the transport files.

Example Code 20.4 Typical FTP Dialog

```
❶ hp> ftp ax7000.vms.sas.com
   Connected to ax7000.vms.com.
   220 ax7000.vms.com MultiNet FTP Server Process V4.0(15) at Thu-Sep 30-99
     12:59PM-EDT
   Name (ax7000.vms.com:): joe
   331 User name (joe) ok. Password, please.
   Password:
   230 User JOE logged into HOSTVAX:[JOE] at Thu 30-Sep-99 12:59PM-EDT, job
     27a34cef.
   Remote system type is VMS.
❷ ftp> cd [.xpttest]
   250 Connected to system-specific file/pathname.
❸ ftp> binary
   200 Type I ok.
❹ ftp> get xptds.dat xptds.dat
   200 Port 14.83 at Host 10.26.2.45 accepted.
   150 IMAGE retrieve of system-specific file/pathname XPTDS.DAT;1 started.
❺ 226 Transfer completed.  1360 (8) bytes transferred.
   1360 bytes received in 0.02 seconds (87.59 Kbytes/s)
❻ ftp> get xptlib.dat xptlib.dat
   200 Port 14.84 at Host 10.26.2.45 accepted.
   150 IMAGE retrieve of system-specific file/pathname XPTLIB.DAT;1 started.
❼ 226 Transfer completed.  3120 (8) bytes transferred.
   3120 bytes received in 0.04 seconds (85.81 Kbytes/s)
❽ ftp> quit
```

1 From the HP-UX target host, the user invokes FTP to connect to the OpenVMS Alpha source host AX7000.VMS.SAS.COM.

2 After a connection is established, at the FTP prompt, user JOE changes to the subdirectory on the source host that contains the transport files.

3 The transport file attribute BINARY indicates that the OpenVMS transport file should be transferred from the source host in BINARY format.

4 The FTP **get** command obtains the transport file named XPTDS.DAT from the source host and copies it to a new file that has the same name, XPTDS.DAT, in the target host current directory.

5 Messages indicate that the transfer was successful and that the size of the transport file was 1360 bytes. Compare the sizes of the transport files at the source host and the target host. If the sizes are identical, then the network successfully transferred the file. For details about listing file size, see “Verifying the Size of a Transport File” on page 161.

6 The FTP **get** command obtains another transport file named XPTLIB.DAT from the source host and copies it to a new file that has the same name, XPTLIB.DAT, in the target host current directory.

7 Messages indicate that the transfer was successful. Compare the sizes of the transport files at the source host and the target host.

8 The user quits the FTP session.

For complete details about using the file transfer utility, see your FTP documentation.

Using PROC COPY at the Target Host to Restore Transport Files into Native Format

The following example shows a SAS program that translates a transport file to native file format.

Example Code 20.5 SAS Program That Restores Transport Files into Native File Format

```
❶ libname xptlib xport 'xptlib.dat';
❷ libname xptds xport 'xptds.dat';

❸ libname natvlib v7 'natvlib';
❹ libname natvds v7 'natvds';

  /* translate transport file for library */
  /* to native format on target host.     */

❺ proc copy in=xptlib out=natvlib;
  run;

  /* translate transport file for data set*/
  /* to native format on target host */

❻ proc copy in=xptds out=natvds;
     select grades;
  run;
```

1 The LIBNAME statement assigns the libref XPTLIB to the physical location XPTLIB.DAT, which stores the entire library that was transferred to the target host. The XPORT engine reads XPTLIB.

2 The LIBNAME statement assigns the libref XPTDS to the physical location XPTDS.DAT, which stores the single data set that was transferred to the target host. The XPORT engine reads XPTDS.

3 The LIBNAME statement assigns the libref NATVLIB to the physical location NATVLIB, which stores the entire library to be translated from transport format to native format. The V7 engine creates NATVLIB.

4 The LIBNAME statement assigns the libref NATVDS to the physical location NATVDS, which stores the single data set to be translated from transport format to native format. The V7 engine creates NATVDS.

5 PROC COPY copies all three data sets from the libref XPTLIB to the new libref NATVLIB. The XPORT engine reads all data sets from XPTLIB in transport format. The V7 engine writes the data sets to the output libref NATVLIB in native HP-UX format.

6 PROC COPY selects the data set GRADES to copy to the new library NATVDS. The XPORT engine reads the data set GRADES in transport format. The V7 engine writes the output library XPTDS in native HP-UX format.

Viewing the SAS Log at the Target Host

The following example shows a SAS log that documents the successful execution of the SAS program shown in "Using PROC COPY at the Target Host to Restore Transport Files into Native Format" on page 139.

Example Code 20.6 Target Host SAS Log

```
NOTE: Copyright (c) 1999 by SAS Institute Inc., Cary, NC, USA.
❶ NOTE: SAS (r) Proprietary Software Version 7 (TS00.00P1D090398)
      Licensed to SAS Institute Inc., Site 0000000001.
❷ NOTE: This session is executing on the HP-UX B.10.20 platform.
NOTE: Running on HP Model 9000/715 Serial Number 2005516582.
libname xptlib xport 'xptlib.dat';
❸ NOTE: Libref XPTLIB was successfully assigned as follows:
      Engine:        XPORT
      Physical Name:  system-specific file/pathname/xptlib.dat
libname xptds xport 'xptds.dat';
❹ NOTE: Libref XPTDS was successfully assigned as follows:
      Engine:        XPORT
      Physical Name:
      system-specific file/pathname/xptds.dat
libname natvlib v7 'natvlib';
❺ NOTE: Libref NATVLIB was successfully assigned as follows:
      Engine:        V7
      Physical Name:
      system-specific file/pathname/natvlib
libname natvds v7 'natvds';
❻ NOTE: Libref NATVDS was successfully assigned as follows:
      Engine:        V7
      Physical Name:
      system-specific file/pathname/natvds

/* translate transport file for library to native */
/* format on target host.                         */
proc copy in=xptlib out=natvlib;
run;
NOTE: Input library XPTLIB is sequential.
❼ NOTE: Copying XPTLIB.GRADES to NATVLIB.GRADES (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set NATVLIB.GRADES has 2 observations and 4 variables.
❽ NOTE: Copying XPTLIB.NUMBERS to NATVLIB.NUMBERS (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set NATVLIB.NUMBERS has 10 observations and 1 variables.
❾ NOTE: Copying XPTLIB.SIMPLE to NATVLIB.SIMPLE (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set NATVLIB.SIMPLE has 1 observations and 3 variables.
/* translate transport file for data set to native */
/* on target host                                  */
proc copy in=xptds out=natvds;
  select grades;
run;
NOTE: Input library XPTDS is sequential.
❿ NOTE: Copying XPTDS.GRADES to NATVDS.GRADES (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set NATVDS.GRADES has 2 observations and 4 variables
```

1 The target host runs SAS Version 7, which means that the SAS session on the target host uses the default library engine V7.

2 The target host runs HP-UX.

3 The LIBNAME statement assigns the libref XPTLIB to the physical device whose specification is platform-dependent. In this example, the physical device indicates an HP-UX operating system. The XPORT engine reads XPTDS.

4 The LIBNAME statement assigns the libref XPTDS to the physical device whose specification is platform-dependent. The XPORT engine reads XPTLIB.

5 The LIBNAME statement assigns the libref NATVLIB to the physical device whose specification is platform-dependent. In this example, the physical device indicates an HP-UX operating system. The V7 engine writes to NATVLIB.

6 The LIBNAME assigns the libref NATVDS to the physical device whose specification is platform-dependent. In this example, the physical device indicates an HP-UX operating system. The V7 engine writes to NATVDS.

7 PROC COPY copies XPTLIB.GRADES to NATVLIB.GRADES. The NATVLIB data set is written in V7 format.

8 PROC COPY copies XPTLIB.NUMBERS to NATVLIB.NUMBERS. The NATVLIB data set is written in V7 format.

9 PROC COPY copies XPTLIB.SIMPLE to NATVLIB.SIMPLE. The NATVLIB data set is written in V7 format.

10 PROC COPY copies XPTLIB.GRADES to NATVDS.GRADES. The NATVDS data set is written in V7 format.

OS/390 to Windows NT File Transport

Using PROC CPORT at the Source Host to Create Transport Files

The following example shows a SAS program that copies two data sets and two catalogs from a library in OS/390 format and writes them to a default output file in transport format.

Example Code 20.7 SAS Program That Copies Data Sets and Catalogs to a Transport File

```
libname test 'joe.mytest.sas';
proc cport library=test;
run;
```

The LIBNAME statement assigns the libref TEST to the physical location JOE.MYTEST.SAS, which points to the library to be transported. JOE is the userid that is associated with the SAS session in which the transport operation is performed. Because no FILE= option is provided, SAS uses the OS/390 default transport file name *userid*.SASCAT.DATA. PROC CPORT reads the contents of the library TEST and writes the library contents to the default output file in transport format.

Viewing the SAS Log at the Source Host

The following example shows a SAS log that documents the successful execution of the SAS program shown in "Using PROC CPORT at the Source Host to Create Transport Files" on page 141.

Example Code 20.8 Source Host SAS Log File

```
libname test 'joe.mytest.sas';
proc cport lib=test;
run;
WARNING: No output file is specified. Default output
 file JOE.SASCAT.DATA is used.

NOTE: Proc CPORT begins to transport data set TEST.CITY
NOTE: The data set contains 7 variables and 72 observations.
NOTE: Transporting data set index information.

NOTE: Proc CPORT begins to transport catalog TEST.FORMATS
NOTE: The catalog has 3 entries
NOTE: Transporting entry REGFMT  .FORMATC
NOTE: Transporting entry SALEFMT .FORMATC
NOTE: Transporting entry SIZEFMT .FORMATC

NOTE: Proc CPORT begins to transport catalog TEST.TEST
NOTE: The catalog has 11 entries
NOTE: Transporting entry ABOUT   .CBT
NOTE: Transporting entry APPEND  .CBT
NOTE: Transporting entry BOOKMENU.CBT
NOTE: Transporting entry DEFAULT .FORM
NOTE: Transporting entry HELP    .HELP
NOTE: Transporting entry CLIST   .LIST
NOTE: Transporting entry ENTRYTYP.LIST
NOTE: Transporting entry SPELLALL.PMENU
NOTE: Transporting entry SPELLSUG.PMENU
NOTE: Transporting entry ADDON1  .PROGRAM
NOTE: Transporting entry ADDON2  .PROGRAM

NOTE: Proc CPORT begins to transport data set TEST.VARNUM
NOTE: The data set contains 10 variables
  and 100 observations.
```

Note: Default output filenames are host-specific. △

PROC CPORT reads the contents of the entire library that is referenced by the libref TEST and writes to the default transport file. The remaining series of notes indicate that PROC CPORT transports the data set TEST.CITY, the catalog TEST.FORMATS, the catalog TEST.TEST, and data set TEST.VARNUM into the transport file JOE.SASCAT.DATA.

Verifying the Transport Files

You are advised to verify the integrity of your transport files at the source host before the files are transferred to the target host. A successful verification at the source host can eliminate the possibility that the transport file was created incorrectly. Also, after you transfer a file to the target host, you can compare the transport file that was sent from the source host with the file that was received at the target host. See "Methods for Verifying Transport Files" on page 160 for details.

Transferring the Transport Files to the Target Host

Verify the file attributes of the transport files before they are transferred to the target host. The following example shows typical output for TSO.

Example Code 20.9 Using TSO LISTD Command to Verify the Attributes of the Transport File

```
listd "userid.xportout.dat"
USERID.XPORTOUT.DAT
--RECFM-LRECL-BLKSIZE-DSORG
  FB    80    8000    PS
--VOLUMES--
  APP009
```

After you verify the attributes of the transport files, you can use FTP to transfer them over the network. Change the default DCB attributes, as necessary, in the FTP dialog. In this example, because the user on the source host has permission to write to the target host, the FTP **put** command is used to write the transport file to the target host.

The following example shows the FTP commands you specify at the source host to write the transport files to the target host.

Example Code 20.10 Typical FTP Dialog

```
 ❶ ftp mypc
 EZA1450I MVS TCP/IP FTP V3R2
 EZA1554I Connecting to SPIDER 10.24.2.32, port 21
 220 spider FTP server (Version 4.162 Tue Nov 1
  10:50:37 PST 1988) ready.
 EZA1459I USER (identify yourself to the host):
userid password
 EZA1701I >>>USER joe
 331 Password required for joe.
 EZA1701I >>>PASS ********
 230 User joe logged in.
 ❷ EZA1460I Command:
binary
 EZA1701I >>>TYPE i
 200 Type set to I.
 ❸ EZA1460I Command:
put 'joe.sascat.data' c:\tport.dat
 ❹ EZA1701I >>>SITE VARrecfm Lrecl=80
  Recfm=FB BLKSIZE=8000
 500 'SITE VARRECFM Lrecl=80 Recfm=FB BLKSIZE=23440':
  EZA1701I >>>PORT 10,253,1,2,129,50
 200 PORT command
 ❺ EZA1701I >>>STOR c:\tport.dat
 150 Opening BINARY mode data connection for c:\tport.dat
 ❻ 226 Transfer complete.
 EZA2517I 6071600 bytes transferred in 13 seconds.
  Transfer rate 466.18 Kbytes/sec.
 ❼ EZA1460I Command:
quit
 EZA1701I >>>QUIT
 221 Goodbye.
 READY
```

1 From the OS/390 source host, the user invokes FTP to connect to the Windows NT target host MYPC.
2 The transport file attribute BINARY indicates that the OS/390 transport file should be transferred from the source host in BINARY format .
3 The FTP **put** command copies the transport file named JOE.SASCAT.DATA from the source host to the target host physical location C:\TPORT.DAT.
4 The FTP file attribute commands indicate a record length of 80 bytes, a fixed record type, and a block size of 8000.
5 TPORT.DAT is saved to drive C.
6 Messages indicate that the transfer was successful. For details about listing a file size, see "Verifying the Size of a Transport File" on page 161.
7 The user quits the FTP session.

Using PROC CIMPORT at the Target Host to Import Transport Files into Native Format

The following example shows a SAS program that translates the transport file from transport format into native format.

Example Code 20.11 SAS Program That Imports Transport Files into Native Format

```
libname newlib 'c:\mylib';
proc cimport infile='c:\tport.dat' library=newlib;
run;
```

This LIBNAME statement assigns the libref NEWLIB to the physical location **c:\mylib**, which stores the entire V7 library. PROC CIMPORT reads the entire content of the transport file that is identified in the INFILE= option and writes it to the output location that is identified in the LIBNAME= option.

As an alternative to importing the entire contents of the library into native V7 format, you can select or exclude specific entities from the transport library.

Here are examples:

Example Code 20.12 Selecting One or More Data Sets

```
filename target 'c:\tport.dat';
libname newlib 'c:\mylib';
proc cimport infile=target library=newlib;
   select varnum;
run;
```

In the preceding example, the fileref TARGET points to the location where the transport file was transferred to the target host. The libref NEWLIB points to the location to store the selected member. PROC CIMPORT reads the entire content of the transport file that is identified in the INFILE= option and writes only the member that is identified in the SELECT statement. The data set VARNUM is written to the library NEWLIB in Windows format.

Example Code 20.13 Selecting a Catalog Entry Type

```
filename target 'c:\tport.dat';
libname newlib 'c:\mylib';
proc cimport infile=target library=newlib
   memtype=catalog et=program;
run;
```

In the preceding example, PROC CIMPORT reads the entire content of the transport file that is identified in the INFILE= option and writes only members of type CATALOG and entries of type PROGRAM to the library NEWLIB in Windows format.

Example Code 20.14 Selecting Catalog Entries

```
filename target 'c:\tport.dat';
libname newlib 'c:\mylib';
proc cimport infile=target library=newlib memtype=cat;
   select spellsug.pmenu addon1.program;
run;
```

In the preceding example, PROC CIMPORT reads the entire content of the transport file that is identified in the INFILE= option and writes only the entries SPELLSUG.PMENU and ADDON1.PROGRAM of member type CATALOG to the library NEWLIB in Windows format.

Viewing the SAS Log at the Target Host

The following example shows a SAS log that documents the successful execution of the SAS program that is shown in "Using PROC CIMPORT at the Target Host to Import Transport Files into Native Format" on page 144.

Example Code 20.15 Target Host Log File

```
NOTE: Proc CIMPORT begins to create/update data set NEWLIB.CITY
NOTE: The data set index REGION is defined.
NOTE: Data set contains 7 variables and 72 observations.

NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FORMATS
NOTE: Entry REGFMT.FORMATC has been imported.
NOTE: Entry SALEFMT.FORMATC has been imported.
NOTE: Entry SIZEFMT.FORMATC has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FORMATS: 3

NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.TEST
NOTE: Entry ABOUT.CBT has been imported.
NOTE: Entry APPEND.CBT has been imported.
NOTE: Entry BOOKMENU.CBT has been imported.
NOTE: Entry DEFAULT.FORM has been imported.
NOTE: Entry HELP.HELP has been imported.
NOTE: Entry CLIST.LIST has been imported.
NOTE: Entry ENTRYTYP.LIST has been imported.
NOTE: Entry SPELLALL.PMENU has been imported.
NOTE: Entry SPELLSUG.PMENU has been imported.
NOTE: Entry ADDON1.PROGRAM has been imported.
NOTE: Entry ADDON2.PROGRAM has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.TEST: 11

NOTE: Proc CIMPORT begins to create/update data set NEWLIB.VARNUM
NOTE: Data set contains 10 variables and 100 observations.
```

PROC CIMPORT creates the data set NEWLIB.CITY, the catalog NEWLIB.FORMAT, the catalog NEWLIB.TEST, and the data set NEWLIB.VARNUM at the target host Windows NT in Windows format.

OS/390 TSO to UNIX File Transport

Using PROC CPORT at the Source Host to Create Transport Files: OS/390

The following example shows a SAS program that creates three data sets in OS/390 format and translates them to transport format.

Example Code 20.16 SAS Program That Creates Data Sets and Transport Files: OS/390

```
❶/* Specify tport file */
filename tport ftp 'tport.dat'
cd='mydir'                    /* Specify directory  */
host='myhost.mycompany.com' /* Specify your host  */
user='myuser'                 /* Specify user       */
pass='mypass'                 /* Specify password   */
rcmd='site umask 022'         /* Set permissions to */
                              /* -rw-r--r--         */
recfm=s                       /* binary transfer    */
debug;                        /* write ftp messages */
 /*-----------------------------------*/
 /* Allocate the SAS test library.    */
 /*-----------------------------------*/
❷ libname trantest '.trantest.lib'
  disp=(new,catlg,delete);

 /*---------------------------------------*/
 /* Creates data set GRADES which         */
 /* contains numeric and character data.  */
 /*---------------------------------------*/
❸ data trantest.grades;
   input student $ test1 test2 final;
datalines;
Fred  66 80 70
Wilma 97 91 98
;
 /*-----------------------------------*/
 /* Creates data set SIMPLE which     */
 /* contains character data only.     */
 /*-----------------------------------*/
 data trantest.simple;
   x='dog';
   y='cat';
   z='fish';
run;

 /*-----------------------------------*/
 /* Creates data set NUMBERS which     */
 /* contains numeric data only         */
 /*-----------------------------------*/
 data trantest.numbers;
  do i=1 to 10;
     output;
```

```
  end;
run;

 /*-----------------------------------*/
 /* Uses PROC CPORT to write the      */
 /* transport file. This transport    */
 /* file is written to UNIX by using  */
 /* the FTP access method.            */
 /*-----------------------------------*/
❹ proc cport library=trantest file=tport;
run;

 /*-----------------------------------*/
 /* Reads the transport file on UNIX  */
 /* by using the FTP access method.   */
 /* Uses PROC CIMPORT to transfer and */
 /* to import the data sets to the    */
 /* WORK library.                     */
 /*-----------------------------------*/
❺ proc cimport infile=tport library=work;
run;
```

1 The FILENAME statement assigns the libref TPORT to the physical location TPORT.DAT, which stores the entire library to be created. The FTP access method option specifies the attributes of the file transfer operation. The RCMD= command and SITE UMASK 022 argument set the user file access permissions on the target host. The RECFM= option and the S argument specify a binary transfer. For details about the FTP access method options, see the syntax for the FILENAME statement in *SAS Language Reference: Dictionary*.

2 The LIBNAME statement assigns TRANTEST to the physical location *syspref*.TRANTEST.LIB, which stores the data sets to be created. The DISP= option specifies the status of the data set at the beginning and the end of the job. NEW indicates that a new data set is to be created. CATLG specifies that the system should place an entry in the system catalog for normal job termination. DELETE specifies that the data set be deleted at the end of the step for abnormal job termination.

3 The next three DATA statements create the SAS data sets.

4 PROC CPORT reads the library data sets from the OS/390 host and writes the transport data to the UNIX target host.

5 PROC CIMPORT reads the transport file from the UNIX target host and writes the SAS data sets to the WORK library.

Viewing the SAS Log at the OS/390 Source Host

The SAS log is presented in three parts. The following example shows the successful creation of data sets on the OS/390 source host.

Example Code 20.17 Source Host OS/390 SAS Log: Part 1 of 3

```
❶ NOTE: SAS (r) Proprietary Software Version 7
  (TS00.00P1P102298) Licensed to SAS INSTITUTE INC.,
  Site 0000000001.
❷ NOTE: This session is executing on the
  OS/390 V02R04M00 platform.
```

```
NOTE: Running on IBM Model 9672,
                  IBM Model 9672,
                  IBM Model 9672.
+++++
❸ filename tport ftp 'tport.dat'/* Specify tport file */
cd='mydir'                      /* Specify directory  */
host='myhost.mycompany.com'     /* Specify your host  */
user='myuser'                   /* Specify user       */
pass='mypass'                   /* Specify password   */
rcmd='site umask 022'           /* Set permissions to */
                                /* -rw-r--r--         */
recfm=s                         /* binary transfer    */
debug;                          /* write ftp messages */
/*-------------------------------------------------*/
/* Allocate the SAS test library.                  */
/*-------------------------------------------------*/
❹ libname trantest '.trantest.lib'
  disp=(new,catlg,delete);
NOTE: Libref TRANTEST was successfully assigned
  as follows:
      Engine:        V7
      Physical Name: JOE.TRANTEST.LIB
/*--------------------------------------*/
/* Creates data set GRADES which        */
/* contains numeric and character data. */
/*--------------------------------------*/
❺ data trantest.grades;
input student $ test1 test2 final;
datalines;
NOTE: The data set TRANTEST.GRADES has
  2 observations and 4 variables.
;
/*-----------------------------------*/
/* Creates data set SIMPLE which     */
/* contains character data only.     */
/*-----------------------------------*/
data trantest.simple;
  x='dog';
  y='cat';
  z='fish';
run;

NOTE: The data set TRANTEST.SIMPLE has
  1 observations and 3 variables.

/*-----------------------------------*/
/* Creates data set NUMBERS which    */
/* contains numeric data only.       */
/*-----------------------------------*/
data trantest.numbers;
  do i=1 to 10;
    output;
  end;
```

```
run;

NOTE: The data set TRANTEST.NUMBERS has
  10 observations and 1 variables.
```

1 The source host runs SAS Version 7, which means that the SAS session default library engine is V7.

2 The IBM source host runs OS/390 V02R04M00.

3 The FILENAME statement identifies the external file and specifies the attributes of the FTP access method, which transfers the transport file over the network.

4 The LIBNAME statement assigns the libref TRANTEST to the location for the files in native format on the OS/390 source host.

5 The data sets TRANTEST.GRADES, TRANTEST.SIMPLE, and TRANTEST.NUMBERS are created in the library that is referenced by TRANTEST.

The following example shows the creation of the transport file and its transfer across the network by using the FTP access method.

Example Code 20.18 OS/390 Source Host SAS Log: Part 2 of 3

```
/*-------------------------------------------*/
/* Uses PROC CPORT to write the library      */
/* TRANTEST data sets to the transport file  */
/* via the FTP access method.                */
/*-------------------------------------------*/
❶ proc cport library=trantest file=tport;
run;
❷ NOTE: 220 myhost FTP server (Version 4.162
  Tue Nov 1 10:50:37 PST 1988) ready.
NOTE: <<< 220 myhost FTP server (Version 4.162
   Tue Nov 1 10:50:37 PST 1988) ready.
NOTE: >>> USER hostftp
NOTE: <<< 331 Password required for hostftp.
NOTE: >>> PASS XXXXXXX
NOTE: <<< 230 User hostftp logged in.
NOTE: >>> PORT 10,253,1,2,30,208
NOTE: <<< 200 PORT command
NOTE: >>> TYPE I
NOTE: <<< 200 Type set to I.
NOTE: >>> CWD transfer
NOTE: <<< 250 CWD command successful.
NOTE: >>> PWD
NOTE: <<< 257 "/mydir" is current directory.
NOTE: >>> site umask 022
NOTE: <<< 200 UMASK set to 022 (was 027)
NOTE: >>> STAT
NOTE: <<< 211- myhost FTP server status:
NOTE: <<<      Version 4.162 Tue
  Nov 1 10:50:37 PST 1988
NOTE: <<<      Connected to sdcmvs.mvs.sas.com
NOTE: <<<      Logged in as hostftp
NOTE: <<<      TYPE: Image; STRUcture: File;
  transfer MODE: Stream
NOTE: <<<      PORT (10,253,1,2,30,208)
```

```
NOTE: <<< 211 End of status
NOTE: >>> STOR tport.dat
NOTE: <<< 150 Opening BINARY mode data connection
  for tport.dat.
NOTE: User hostftp has connected to FTP server
  on Host myhost.unx.com.

NOTE: Proc CPORT begins to transport
  data set TRANTEST.GRADES
NOTE: The data set contains 4 variables and
  2 observations. Logical record length is 32.

NOTE: Proc CPORT begins to transport
  data set TRANTEST.NUMBERS
NOTE: The data set contains 1 variables
  and 10 observations.
      Logical record length is 8.

NOTE: Proc CPORT begins to transport
   data set TRANTEST.SIMPLE

NOTE: The data set contains 3 variables
  and 1 observations.
      Logical record length is 10.
NOTE: <<< 226 Transfer complete.
NOTE: >>> QUIT
```

1 PROC CPORT copies the files that are identified in the LIBRARY= option (from OS/390 format) to the library that is identified in the FILE= option (in transport format).

2 The FTP access method connects to the UNIX target host across the network and writes the file referenced by the libref TPORT to the UNIX target host.

The following example shows the successful import of the transport file to native format on the UNIX target host and a verification of the method that was used for creating the transport file on the OS/390 source host.

Example Code 20.19 OS/390 Source Host SAS Log: Part 3 of 3

```
/*-------------------------------------------*/
/* Use FTP access method to read the transport */
/* file. Use PROC CIMPORT to import the test   */
/* data sets to library WORK.                  */
/*-------------------------------------------*/
❶ proc cimport infile=tport library=work;
run;

❷ NOTE: 220 myhost FTP server (Version 4.162
  Tue Nov 1 10:50:37 PST 1988) ready.
NOTE: <<< 220 myhost FTP server (Version 4.162
  Tue Nov 1 10:50:37 PST 1988) ready.
NOTE: >>> USER hostftp
NOTE: <<< 331 Password required for hostftp.
NOTE: >>> PASS XXXXXXX
NOTE: <<< 230 User hostftp logged in.
NOTE: >>> PORT 10,253,1,2,30,210
```

```
NOTE: <<< 200 PORT command
NOTE: >>> TYPE I
NOTE: <<< 200 Type set to I.
NOTE: >>> CWD transfer
NOTE: <<< 250 CWD command successful.
NOTE: >>> PWD
NOTE: <<< 257 "/mydir" is current directory.
NOTE: >>> site umask 022
NOTE: <<< 200 UMASK set to 022 (was 027)
NOTE: >>> STAT
NOTE: <<< 211- myhost FTP server status:
NOTE: <<<   Version 4.162 Tue Nov 1 10:50:37 PST 1988
NOTE: <<<   Connected to sdcmvs.mvs.sas.com
NOTE: <<<   Logged in as hostftp
NOTE: <<<   TYPE: Image; STRUcture: File; transfer
NOTE: <<<   MODE: Stream PORT (10,253,1,2,30,210)
NOTE: <<< 211 End of status
NOTE: >>> RETR tport.dat
NOTE: <<< 150 Opening BINARY mode data connection for
  tport.dat (2320 bytes).
NOTE: User hostftp has connected to FTP server on
  Host myhost.unx.com .
NOTE: Proc CIMPORT begins to create/update data set
  WORK.GRADES
NOTE: Data set contains 4 variables and
  2 observations.Logical record length is 32

NOTE: Proc CIMPORT begins to create/update
  data set WORK.NUMBERS
NOTE: Data set contains 1 variables
 and 10 observations. Logical record length is 8

NOTE: Proc CIMPORT begins to create/update
 data set WORK.SIMPLE

NOTE: Data set contains 3 variables and
  1 observations.
      Logical record length is 10

NOTE: <<< 226 Transfer complete.
NOTE: >>> QUIT
```

1 PROC CIMPORT copies the file that is identified in the INFILE= option (from transport format) to the library that is identified in the LIBRARY= option (in native OS/390 format).

2 To validate the accuracy of the method that was used for importing the transport file into native format on the UNIX target host, the FTP access method returns the transport file TPORT.DAT to the OS/390 source host across the network. On the local host, PROC CIMPORT copies the transport file into native format. A successful execution of PROC CIMPORT on the source host verifies that a valid transport file was created at the UNIX target host.

OS/390 JCL Batch to UNIX File Transport

The OS/390 JCL Batch Program

Although presented in four parts, the following program is designed as a single program. The parts perform these tasks:

1. Use PROC COPY to create a transport file on the OS/390 source host.
2. Transfer the transport file over the network to the UNIX target host.
3. Verify the accuracy of the transport file.
4. Use PROC COPY to restore the transport file back to the OS/390 source host.

Embedded comments document the program.

Using PROC COPY to Create a Transport File

The following example shows the first part of the program that creates three data sets in OS/390 format and translates them to transport format. For details in the SAS log that documents the execution of this program part, see “Recording the Creation of Data Sets and Transport Files in the SAS Log” on page 156.

Example Code 20.20 Creating Data Sets and Transport Files

```
//XPORTTST JOB job-card-information
//*-------------------------------------------
//* Run SAS step that creates a transport library
//* for the three SAS test data sets.

//*-------------------------------------------
//SASOUT    EXEC SAS
//*-------------------------------------------
//* Allocate the SAS XPORTOUT library.
//* The XPORTOUT library should have the
//* following data set information:
//* Record format: FB
//* Record length: 80
//* Block size:    8000
//* Organization:  PS

//*-------------------------------------------

//XPORTOUT DD DSN=userid.XPORTOUT.DAT, DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//            SPACE=(TRK,(1,1))
//SYSIN    DD *
 /*-----------------------------------------*/
 /* Assign the SAS test xport library       */
 /*-----------------------------------------*/
libname xportout xport;

 /*-----------------------------------------*/
 /* Creates data set GRADES which contains  */
 /* numeric and character data.             */
```

```
 /*---------------------------------------*/
data grades;
   input student $ test1 test2 final;
datalines;
Fred  66 80 70
Wilma 97 91 98
;

 /*-----------------------------------*/
 /* Creates data set SIMPLE which     */
 /* contains character data only.     */
 /*-----------------------------------*/
data simple;
   x='dog';
   y='cat';
   z='fish';
run;

 /*-----------------------------------*/
 /* Creates data set NUMBERS which    */
 /* contains numeric data only.       */
 /*-----------------------------------*/
data numbers;
  do i=1 to 10;
     output;
  end;
run;
 /*-----------------------------------*/
 /* Copy the three test data sets to  */
 /* the XPORT library.                */
 /*-----------------------------------*/
proc copy in=work out=xportout;
run;
/*
```

Transferring the Transport File across the Network

The following example shows the generation of the FTP command file and the transfer of the transport file over the network to the target host. For details in the SAS log that documents the execution of this program part, see "Recording the Transfer of the Transport File to the Target Host in the SAS Log" on page 157.

Example Code 20.21 Using FTP to Transfer Transport Files

```
//*-------------------------------------------------
//* Generate FTP command file for sending XPORTOUT
//* test library to the target host.
//*-------------------------------------------------
//FTPCMDO  EXEC PGM=IEBGENER,COND=EVEN
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=userid.FTP.OUT,
//            UNIT=DISK,DISP=(NEW,CATLG),
//            SPACE=(TRK,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
```

```
//*-----------------------------------------------
//* Ensure that the FTP commands specify a BINARY
//* mode transfer.
//*-----------------------------------------------
//SYSUT1   DD *
userid password
cd mydir
binary
put 'userid.xportout.dat' xportout.dat
quit
/*
//*-----------------------------------------------
//* FTP library XPORTOUT to the target host.
//*-----------------------------------------------
//FTPXEQO EXEC PGM=IKJEFT01,REGION=2048K,DYNAMNBR=50,COND=EVEN
//SYSPRINT DD SYSOUT=*
//SYSTSOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ALLOC FI(input) DA('userid.FTP.OUT') SHR
FTP target-host (EXIT
/*
```

Verifying the Accuracy of the Transport File

The following example shows the verification of the transport file by transferring it from the UNIX target host to the OS/390 source host in native format. A successful translation from transport format to native OS/390 format verifies the accuracy of the transport file. For details in the SAS log that document the execution of this program part, see "Recording the Verification of the Transport File in the SAS Log" on page 158.

Example Code 20.22 Verifying Transport Files

```
//*-----------------------------------------------
//* The following steps retrieve the XPORTOUT library
//* from the target host and read the three test
//* data sets back into the WORK library.
//*-----------------------------------------------
//* Generates the FTP command file for getting
//* the test library XPORTOUT from the target host.
//*-----------------------------------------------
//FTPCMDI  EXEC PGM=IEBGENER,COND=EVEN
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=userid.FTP.IN,
//            UNIT=DISK,DISP=(NEW,CATLG),
//            SPACE=(TRK,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//*-----------------------------------------------
//* The FTP commands specify a BINARY mode
//* transfer.  Uses the LOCSITE command to define
//* the correct XPORT library data set information.
//*-----------------------------------------------
//SYSUT1   DD *
userid password
```

```
cd mydir
locsite recfm=fb blocksize=8000 lrecl=80
binary
get xportout.dat 'userid.xportin.dat'
quit
/*
//*-----------------------------------------------
//* Connects to the target host and retrieves
//* the library XPORTOUT.
//*-----------------------------------------------
//FTPXEQI  EXEC PGM=IKJEFT01,REGION=2048K,DYNAMNBR=50,COND=EVEN
//SYSPRINT DD SYSOUT=*
//SYSTSOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ALLOC FI(input) DA('userid.FTP.IN') SHR
FTP target-host (EXIT
/*
```

Using PROC COPY to Restore the Transport File

The following example restores the transport file to native format on the OS/390 source host. For details in the SAS log that document the execution of this program part, see "Recording the Restoration of the Transport File to the Source Host in the SAS Log" on page 159.

Example Code 20.23 Restoring the Transport File to Native Format

```
//*-----------------------------------------------
//* Runs SAS step that reads the transport library
//* and writes the three SAS test data sets to
//* library WORK.

//*-----------------------------------------------
//SASIN     EXEC SAS
//XPORTIN   DD DSN=userid.XPORTIN.DAT,DISP=SHR
//SYSIN    DD *
 /*-----------------------------------------------*/
 /* Assigns the SAS test library XPORTIN.        */
 /*-----------------------------------------------*/
libname xportin xport;

 /*-----------------------------------------------*/
 /* Reads the transport file and writes the test */
 /* data sets to library WORK.                   */
 /*-----------------------------------------------*/

proc copy in=xportin out=work;
run;
/*
```

Recording the Creation of Data Sets and Transport Files in the SAS Log

The following example shows the SAS log that documents the creation of the data sets and corresponding transport files.

Example Code 20.24 Viewing the SAS Log at the OS/390 Source Host (Part 1 of 4)

```
                      The SAS System
                      11:03 Monday, October 26, 1999

NOTE: Copyright (c) 1999 by SAS Institute Inc.,
 Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Version 6.09.0460P0304986
       Licensed to SAS INSTITUTE INC., Site 0000000001.

NOTE: Running on IBM Model 9672,
                 IBM Model 9672,
                 IBM Model 9672.

NOTE: No options specified.

/*-----------------------------------------------*/
/* Assigns the SAS test library XPORTOUT.       */
/*-----------------------------------------------*/
libname xportout xport;
NOTE: Libref XPORTOUT was successfully assigned
  as follows:
      Engine:        XPORT
      Physical Name: JOE.XPORTOUT.DAT

/*-----------------------------------------------*/
/* Creates data set GRADES which contains        */
/* numeric and character data.                   */
/*-----------------------------------------------*/
data grades;
   input student $ test1 test2 final;
datalines;

NOTE: The data set WORK.GRADES has 2 observations
  and 4 variables.

/*-------------------------------------*/
/* Creates data set SIMPLE which       */
/* contains character data only.       */
/*-------------------------------------*/
data simple;
x='dog';
y='cat';
z='fish';
run;

NOTE: The data set WORK.SIMPLE has
 1 observations and 3 variables.
```

```
/*-----------------------------------*/
/* Creates data set NUMBERS which    */
/* contains numeric data only.       */
/*-----------------------------------*/
data numbers;
   do i=1 to 10;
   output;
   end;
run;
NOTE: The data set WORK.NUMBERS has
 10 observations and 1 variables.
/*-----------------------------------*/
/* Copies the three test data sets to */
/* the XPORTOUT library.             */
/*-----------------------------------*/
proc copy in=work out=xportout;
run;

NOTE: Copying WORK.GRADES to XPORTOUT.GRADES
 (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPORTOUT.GRADES has
 2 observations and 4 variables.
NOTE: Copying WORK.NUMBERS to XPORTOUT.NUMBERS
(MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPORTOUT.NUMBERS has
 10 observations and 1 variables.
NOTE: Copying WORK.SIMPLE to XPORTOUT.SIMPLE
(MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set XPORTOUT.SIMPLE has
  1 observations and 3 variables.
```

Note: The notes about SAS system option BUFSIZE do not indicate an error condition. BUFSIZE specifies the permanent buffer size for an output data set, which can be adjusted to improve system performance. The system value that is assigned to the BUFSIZE option is used because the XPORT engine does not support the BUFSIZE= option. See your operating environment companion for details. △

Recording the Transfer of the Transport File to the Target Host in the SAS Log

The following example shows the SAS log that documents the transfer of the transport file to the target host.

Example Code 20.25 Vewing the SAS Log at the OS/390 Source Host (Part 2 of 4)

```
EZA1450I MVS TCP/IP FTP V3R2
EZA1772I FTP: EXIT has been set.
EZA1736I conn MYHOST.MYCOMPANY.COM
EZA1554I Connecting to MYHOST.MYCOMPANY.COM
```

```
  10.26.11.235, port 21
220 myhost FTP server (Version 4.162 Tue Nov 1 10:50:37 PST 1988)
  ready.
EZA1459I USER (identify yourself to the host):
EZA1701I >>>USER joe
331 Password required for joe.
EZA1701I >>>PASS ********
230 User joe logged in.
EZA1460I Command:
EZA1736I cd joe
EZA1701I >>>CWD joe
250 CWD command successful.
EZA1460I Command:
EZA1736I binary
EZA1701I >>>TYPE i
200 Type set to I.
EZA1460I Command:
EZA1736I put 'joe.xportout.dat'
  xportout.dat
EZA1701I >>>SITE VARrecfm Lrecl=80
  Recfm=FB BLKSIZE=8000
500 'SITE VARrecfm Lrecl=80 Recfm=FB
  BLKSIZE=8000': command not understood
EZA1701I >>>PORT 10,253,1,2,33,182
200 PORT command.
EZA1701I >>>STOR xportout.dat
150 Opening BINARY mode data connection for
  xportout.dat.
226 Transfer complete.
EZA1460I Command:
EZA1736I quit
EZA1701I >>>QUIT
```

Recording the Verification of the Transport File in the SAS Log

The following example shows the SAS log that documents the portion of the program that verifies the accuracy of the transport files that were transferred.

Example Code 20.26 Viewing the SAS Log at the OS/390 Source Host (Part 3 of 4)

```
EZA1450I MVS TCP/IP FTP V3R2
EZA1772I FTP: EXIT has been set.
EZA1736I conn MYHOST.MYCOMPANY.COM
EZA1554I Connecting to MYHOST.MYCOMPANY.COM
   10.26.11.235, port 21
220 myhost FTP server (Version 4.162 Tue Nov 1 10:50:37 PST 1988)
   ready.
EZA1459I USER (identify yourself to the host):
EZA1701I >>>USER joe
331 Password required for joe.
EZA1701I >>>PASS ********
230 User joe logged in.
EZA1460I Command:
EZA1736I cd joe
```

```
EZA1701I >>>CWD joe
250 CWD command successful.
EZA1460I Command:
EZA1736I locsite recfm=fb blocksize=8000 lrecl=80
EZA1460I Command:
EZA1736I binary
EZA1701I >>>TYPE i
200 Type set to I.
EZA1460I Command:
EZA1736I get xportout.dat 'joe.xportin.dat'
EZA1701I >>>PORT 10,253,1,2,33,184
200 PORT command
EZA1701I >>>RETR xportout.dat
150 Opening BINARY mode data connection for
  xportout.dat(3120 bytes).
226 Transfer complete.
EZA1617I 3120 bytes transferred in 0.198 seconds.Transfer rate
  9.12 Kbytes/sec.
EZA1460I Command:
EZA1736I quit
EZA1701I >>>QUIT
```

Recording the Restoration of the Transport File to the Source Host in the SAS Log

The following example shows the SAS log that documents the portion of the program that copies the transport file to native format on the OS/390 host.

Example Code 20.27 Viewing the SAS Log at the OS/390 Source Host (Part 4 of 4)

```
NOTE: SAS (r) Proprietary Software Release 6.09.0460P030498
       Licensed to SAS INSTITUTE INC., Site 0000000001.
NOTE: Running on IBM Model 9672,
                 IBM Model 9672,
                 IBM Model 9672.

NOTE: No options specified.

/*----------------------------------------*/
/* Assigns the SAS test library XPORTIN. */
/*----------------------------------------*/
libname xportin xport;
NOTE: Libref XPORTIN was successfully assigned
  as follows:
      Engine:        XPORT
      Physical Name: JOE.XPORTIN.DAT
/*-----------------------------------------------*/
/* Reads the transport file and writes the     */
/* test data sets to the library WORK.         */
/*-----------------------------------------------*/
proc copy in=xportin out=work;
run;

NOTE: Input library XPORTIN is sequential.
```

```
NOTE: Copying XPORTIN.GRADES to WORK.GRADES
  (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across
  different engines. System Option for BUFSIZE was used.
NOTE: The data set WORK.GRADES has 2 observations
  and 4 variables.
NOTE: Copying XPORTIN.NUMBERS to WORK.NUMBERS
  (MEMTYPE=DATA).
NOTE: BUFSIZE is not cloned when copying across
  different engines. System Option for BUFSIZE was used.
NOTE: The data set WORK.NUMBERS has 10 observations
   and 1 variables.
NOTE: Copying XPORTIN.SIMPLE to WORK.SIMPLE
  (MEMTYPE=DATA).
```

Note: The notes about the SAS system option BUFSIZE do not indicate an error condition. BUFSIZE specifies the permanent buffer size for an output data set, which can be adjusted to improve system performance. The system value that is assigned to the BUFSIZE option is used because the XPORT engine does not support the BUFSIZE= option. See your operating environment companion documentation for details. △

Methods for Verifying Transport Files

Here are several suggestions for verifying transport files. The first two methods are performed before and after the file transfer. The final method is performed after the file transfer.

- Restore the transport file back to the source host into native format.
- Use the operating system's list command to verify that a transport file was successfully created.
- For data sets only, use PROC CONTENTS to compare the content of the original data set that was created on the source host with the file that was restored on the target host.

Restoring the Transport File at the Source Host

Use the appropriate method (PROC COPY or PROC CIMPORT) to restore the transport file back to your source host. A successful translation of the transport file to native format on the source host verifies the integrity of the transport file to be transferred.

This example shows the creation of a transport file:

```
libname xptlib xport 'xptlib.dat';
/* create a transport file for the entire library */
proc copy in=work out=xptlib;
run;
```

PROC COPY reads the library from the libref WORK and writes the transport file to the libref XPTLIB.

This example restores the transport file just created back to the source host:

```
libname test 'test';
/* restore the transport file at the source host */
proc copy in=xptlib out=test;
```

```
run;
```

The value for the OUT= option in the example that creates the transport file becomes the value for the IN= option in the example that restores the transport file back to the source host. To protect against overwriting the original data library that is created in WORK, direct output to the library TEST. PROC COPY reads the transport file from the libref XPTLIB and restores it to the libref TEST in native format.

For complete details about the syntax for these procedures, see *SAS Procedures Guide*.

Verify the outcome of this test by viewing the SAS log at the source host. If the transport operation succeeded at the source host, then you can assume that the transport file content is correct. If the transport operation failed, then you can assume that the transport file was not created correctly. In this case, re-create the transport file and restore it again at the source host.

Verifying the Size of a Transport File

Use your operating system's list command to verify that the transport file was successfully created. Here is an OpenVMS Alpha example:

```
vms> dir/size=all *dat

     Directory HOSTVAX:[JOE.XPTTEST]

     XPTDS.DAT;1                7/8
     XPTLIB.DAT;1               7/8
```

The sizes of both files are 7/8 of a block, which is equivalent to 448 bytes.

Here is a UNIX example:

```
$ ls -l *dat
-rw-r--r-- 1 joe    mkt   448 Oct 13 14:24 xptds.dat
-rw-r--r-- 1 joe    mkt   890 Oct 13 14:24 xptlib.dat
```

The size of XPTDS.DAT is 448 bytes; XPTLIB.DAT, 890 bytes.

The method for listing a file size varies according to host.

Compare the size of the transport file on the source host with the size of the transport file that is transferred to the target host. If the sizes of the transport files are identical, then you can assume that the network successfully transferred these files. If the sizes are not the same, you can assume that the network transfer failed. In this case, review the transfer options and re-try the transfer.

Comparing the Original Data Set with the Restored Data Set

You can use the CONTENTS procedure to reveal discrepancies between the original data set at the source host and the restored data set at the target host. A comparison could reveal a misconception about the transported data. For example, upon examination of the data set, you could learn that an entire library of data sets was mistakenly transported rather than only the intended data set.

Use the CONTENTS procedure or the PRINT procedure to list the contents of members of type DATA.

In this example, PROC CONTENTS shows the contents of a single data set in a library:

Example Code 20.28 Using PROC CONTENTS to Show the Contents of a Data Set

```
proc contents data=xptds._all_;
                          CONTENTS PROCEDURE

          Data Set Name: XPTDS.GRADES          Observations:          .
          Member Type:   DATA                  Variables:             4
          Engine:        XPORT                 Indexes:               0
          Created:       .                     Observation Length:    32
          Last Modified: .                     Deleted Observations:  0
          Protection:                          Compressed:            NO
          Data Set Type:                       Sorted:                NO
          Label:

              -----Alphabetic List of Variables and Attributes-----

                       #    Variable    Type    Len    Pos
                       -----------------------------------
                       4    FINAL       Num       8     24
                       1    STUDENT     Char      8      0
                       2    TEST1       Num       8      8
                       3    TEST2       Num       8     16
                  DATAPROG: Creates datasets for TRANSPORTING

                              CONTENTS PROCEDURE

                             -----Directory-----

           Libref:        XPTDS
           Engine:        XPORT
           Physical Name: $1$DUA330:[HOSTVAX.JOE.XPTTEST]XPTDS.DAT

                          #  Name    Memtype  Indexes
                          --------------------------
                          1  GRADES  DATA
```

If you detect problems, re-create the transport file and restore it again at the source host.

Glossary

Accessing a SAS file
is the process whereby a user is permitted to read, write, or update a SAS file on a remote host across a network. Such a user does not typically own the file.

Architectural compatibility
is a characteristic shared by two or more hosts that use identical internal representations for storing numeric data or character data or both. Compatible hosts share identical representations along these dimensions: Floating point numeric storage (IEEE and IBM 390); Character encoding (ASCII or EBCDIC), Endian (big or little); Word alignment (4-byte boundaries or 8-byte boundaries), or Data type length (16-bit, 32-bit, or 64-bit).

Host compatibility allows the exchange of data with no loss of precision or accuracy. In order to exchange data between incompatible hosts, the SAS file first must be converted to transport format. See Transport format.

Backward compatibility
is the ability of a SAS client that runs a later SAS version (such as Version 7 or Version 8) to read, write, and update a SAS file that was created at an earlier SAS version (such as Version 6) as long as the client's application does not implement new features such as long names. The SAS client and application that run the later version are said to be backward compatible with the SAS file that runs the earlier version. See also Forward compatibility.

Binary file
is a file stored in binary format, which cannot be text edited. Binary files are usually executable, but they can contain data only.

Catalog
See SAS catalog.

Catalog entry
See SAS catalog entry.

CEDA (Cross-Environment Data Access)
is a feature of SAS/CONNECT that enables hosts of any type to create a SAS file in non-native format and to read a SAS file, regardless of the compatibility of the host's internal representation with the format of the file being accessed. Using the file's universal header, any type of accessing host can dynamically translate the file to the format that is native to the host for reading only. However, in order for the accessing

host to write to or update the file, the file's format already must be compatible with that of the accessing host or the file's owner must assign to the file a format that is native to the accessing host. See also Native host format and Foreign host format.

Client
See SAS/SHARE client.

Communications access method
is the method your local session uses to communicate with a remote host. Values for the communications access method are specified with the COMAMID= system option.

Compatible hosts
See Architectural compatibility.

Converting a SAS file
is the process of changing the format of a SAS file to that used by the SAS version that runs on the target host. After a SAS file has been copied to the target host, convert it by using the COPY procedure with options that identify the later version input file and the earlier version output file as defined by the LIBNAME statements. See also Copying a SAS file and Target host.

Copying a SAS file
is the process of transferring a SAS file between compatible hosts either by way of a magnetic medium or across a network. No transporting or converting is performed. To access the copied file, use the LIBNAME statement to assign a libref to the file. See also Converting a SAS file, Moving a SAS file, and Transporting a SAS file.

Corrupt file
is the result of an operation that alters the file's data or the file's header, causing the file's structure or its contents to be inaccessible. A common cause of corruption during file transport is that the transport file contains one or more incorrectly placed carriage returns or line feeds to mark the end of record, which makes the entire file unreadable after it is transferred across a network. Invalid file attributes set through the communications software can cause corruption.

Cross-Environment Data Access
See CEDA.

Cross-version environment
is one in which SAS clients and servers use different versions or releases of SAS software. These factors control whether a SAS file can be accessed for reading, writing, or updating: 1) the version of SAS run by the server, 2) the version of SAS run by the client, 3) the version of SAS that was used to create the file being accessed, and 4) the member accessed.

Data control block (DCB)
is the OS/390 control block that contains information about the physical characteristics of an operating system data set.

Data file
See SAS data file.

Data precision
is the reliability of numeric data in a SAS file that is exchanged between hosts. Compatible hosts, which store floating-point numeric data identically, exchange precise numeric data. Precision is lost when passed between incompatible hosts. The two primary floating-point numeric formats are IEEE and IBM 390. See also Architectural compatibility.

Data set
See SAS data set.

Data view
See SAS data view.

Engine
is a part of the SAS System that reads from or writes to a file. Each engine allows the SAS system to access files with a particular format. There are several types of engines, including V6, V7, and V8. See also V6 engine, V7 engine, and V8 engine.

External file
is a file maintained by the host operating system that the SAS System can read data from and route output to. External files can contain raw data, SAS programming statements, procedures output, or output created by the PUT statement. An external file is not a SAS data set. See also Fileref.

FAT-style disk drive
is a File Allocation Table that is maintained on disk drives that are used exclusively with Windows and OS/2 hosts. The FAT file system keeps track of the status of various segments of disk space used for file storage.

FDB (Financial Database)
is a variation of the SAS Multidimensional Database (MDDB) that has been enhanced for use by CFO Vision. See also MDDB.

Fileref
is a name temporarily assigned to an external file or to an aggregate storage location that identifies it to the SAS System. You assign a fileref with a FILENAME statement or with an operating system command.

Do not confuse filerefs with librefs. Filerefs are used for external files; librefs are used for SAS data libraries. See also Libref.

File transfer protocol
See FTP.

Financial Database
See FDB.

Foreign host format
is a relative term that contrasts the format of the file being accessed with the internal data representation of the host that is accessing the file. If the internal formats of the remote host and the file being accessed are not the same, then the remote host can read, but cannot write to or update the file. A foreign host format is also referred to as a non-native or an alien host format.

As an example, the format file created by a mainframe is considered foreign to that of a Windows host. However, the format of a Windows host that is attempting to access a file that is in Windows format is not foreign. See also Native file format.

Forward compatibility
is the ability of a SAS client that runs an earlier version of SAS to read, write, and update a SAS file that was created with a later version of SAS as long as the SAS file does not implement features that are specific to the later version, such as long names. The accessing SAS client and the application that run the earlier version of SAS are said to be forward compatible with the SAS file being accessed that was created with a later version of SAS. See also Backward compatibility.

FTP (File Transfer Protocol)
In TCP/IP, FTP is an application protocol used for transferring files to and from hosts across a network. FTP requires a userid and usually a password to allow access to the remote host.

Generation data sets
are historical copies of a SAS data set. Multiple copies of a SAS data set can be kept by requesting the generations feature. The multiple copies represent versions of the

same data set, which are archived each time it is replaced. The copies are referred to as a generation group and are a collection of data sets that have the same root member name but different version numbers. There is a base version, which is the most recent version, plus a set of historical versions.

Import
is to restore a SAS transport file to its original form (a SAS data library, a SAS catalog, or a SAS data set) in the format appropriate to the target host operating system. Use the CIMPORT procedure to restore a SAS transport file created by the CPORT procedure. Import refers specifically to the use of the CIMPORT procedure for transport file restoration. In general, this term also includes the use of the COPY procedure with the XPORT engine for transport file restoration. See also Restoring a transport file.

Incompatible hosts
See Architectural compatibility.

Integrity constraints
are a set of data validation rules that are specified to restrict the data values accepted into a SAS data file. Using them can preserve the correctness and consistency of stored data. SAS enforces the integrity constraints each time data is inserted or updated in a variable that contains integrity constraints.

Item store
is a SAS member type that is a hierarchical file system with advanced performance features. An item store can contain store registry information and ODS templates, for example. Item stores on the OS/390 host also can contain HTML help files. SAS item stores are of member type ITEMSTOR.

Job control language (JCL)
is a language used to communicate information about a job to the operating system, including the data sets, time, and memory that the job needs.

Library concatenation
is the logical combination of two or more libraries that allows access to the SAS data sets in the combined libraries by using a single libref.

Library reference
See libref.

Libref
is the name temporarily associated with a SAS data library. For example, in the name SASUSER.ACCOUNTS, the name SASUSER is the libref. You can assign a libref with a LIBNAME statement or with operating system control language.

Local SAS session
is a SAS session running on the local host. The local session accepts SAS statements and passes those that are remote submitted to the remote host for processing. The local session manages the output and messages from both the local session and the remote session.

Long names
is a Version 7 and later enhancement to SAS that extends the maximum length of names from the lengths defined in Version 6. This enhancement applies to the names of variables, data sets, procedures, options, statement labels, and librefs or filerefs. Maximum lengths for long names vary according to the type of name. Truncation rules are applied to long names when regressing a Version 7 or later file to a Version 6 host.

MDDB (Multidimensional Database)
is a SAS database format for summary tables that optimizes access and retrieval times for the data that the tables contain by storing data in pre-summarized format.

Member type
is a name assigned by the SAS System that identifies the type of information stored in a SAS file. Member types include ACCESS, CATALOG, DATA, FDB, ITEMSTOR, MDDB, PROGRAM, and VIEW.

Migrating SAS files
is the process of moving SAS files (data and applications) from a host that runs an earlier version of SAS to another host that runs a later version of SAS in order to take advantage of features from the later version. See also Moving a SAS file.

Mixed library
is a Version 7 and later library that contains both Version 7 and later SAS files and Version 6 SAS files. Although mixed libraries are permitted, their maintenance can be difficult. To access a specific Version 6 file or Version 7 or later file in a Version 7 or later library, use the LIBNAME statement with the appropriate engine option. To access all Version 6 files in a mixed library, specify explicitly the V6 engine in the LIBNAME statement. To access all Version 7 or later files in a mixed library, specify explicitly the V7 or V8 engine in the LIBNAME statement. If the engine option is omitted from the LIBNAME statement, the base engine that is used by the SAS session is selected, by default. See also SAS filename extension, V6 engine, V7 engine, and V8 engine.

Moving a SAS file
is the process of passing a SAS file from one host to another host either by way of a magnetic medium or across a network. Three specific variations on moving a SAS file are converting, copying, and transporting. See Converting a SAS file, Copying a SAS file, and Transporting a SAS file.

Multidimensional Database
See MDDB.

Native host format
is a relative term that compares the format of the SAS file being accessed with the internal data representation that is used by the remote host. If the remote host and the file being accessed share a common internal data representation, then the remote host can read, write, and update the file. As an example, a Windows host has read, write, and update access to a file that is in Windows format. See also Foreign host format.

Regressing a file
is the process of moving a SAS file from a later version to an earlier version of SAS; for example, from Version 8 to Release 6.12. Reverting a file to an earlier version requires the setting of the VALIDVARNAME option to the selected engine; for example, OPTIONS VALIDVARNAME=V6, prior to using PROC COPY. If the file created in the later version contains features that do not exist in the earlier version, such as integrity constraints, then you cannot regress the file. Instead, you re-create the file on a host that runs the later version of SAS.

Remote host
in SAS/CONNECT software, is the computer on which processing occurs when you execute a PROC DOWNLOAD, PROC UPLOAD, or other SAS statement that is executed with the RSUBMIT command or the RSUBMIT statement. The term “remote” describes how you interact with the SAS session running on the computer; it is not related to the physical location of the computer. See also Local SAS session.

Restoring a transport file
is the activity that returns the SAS transport file to its original form (a SAS data library, a SAS catalog, or a SAS data set) in the format appropriate to the target host operating system. Restoration is performed using either of two techniques, as

appropriate: 1) the COPY procedure to restore a SAS transport file created by the COPY procedure with the XPORT engine, 2) the CIMPORT procedure to restore a SAS transport file created by the CPORT procedure. Also referred to as reading or importing a transport file. See also Import.

SAS catalog
is a SAS file that stores many different kinds of information in smaller units called catalog entries. A single SAS catalog can contain several different types of catalog entries. Some catalog entries contain system information such as key definitions. Other catalog entries contain application information such as window definitions, help windows, formats, informats, macros, or graphics output. See also SAS catalog entry.

SAS catalog entry
is a separate storage unit within a SAS catalog. Each entry has an entry type that identifies its purpose to the SAS System.

SAS data file
is a SAS data set that is implemented in a form that contains both the data values and the descriptor information. SAS data files have the type DATA.

SAS data library
is a collection of SAS files accessed by the same library engine and recognized as a logical unit by the SAS System. Each file is a member of the library.

SAS data set
is descriptor information and its related data values organized as a table of observations and variables that can be processed by SAS software. A SAS data set can be either a SAS data file or a SAS data view. See also SAS data file and SAS data view.

SAS data view
is a SAS data set in which the descriptor information and the observations are obtained from other files. A SAS data view contains only the descriptor and other information required to retrieve the data values from other SAS files. Both PROC SQL views and SAS/ACCESS views are considered SAS data views. SAS data views are of member type VIEW.

SAS filename extension
is a standard filename identifier that captures these file attributes: 1) the engine that was used to create the file, 2) the architecture of the host on which the file was created, and 3) the member type in the file. SAS uses filename extensions as a key for identifying the appropriate files for access. The length of a Version 6 filename extension varies according to the host, whereas a Version 7 and later filename extension is limited to eight characters. As an example, filename extension **`.sas7bdat`** always identifies a file that was created with the V7 or later engine on a UNIX host for a member of type DATA. See also Architectural compatibility, SAS member type, V6 engine, V7 engine, and V8 engine.

SAS/SHARE client
is a SAS session that requests access to remote data by means of a SAS/SHARE server.

SAS/SHARE server
is the result of an execution of the SERVER procedure. The SERVER procedure is part of SAS/SHARE software. A SAS/SHARE server is also referred to as a server or a SAS server. A server runs in a separate SAS execution that services users' SAS sessions by controlling and executing input and output requests to one or more SAS data libraries.

Server
See SAS/SHARE server.

Source host
is the host from which a SAS file is moved.

Target host
is the host to which a SAS file is moved.

Transferring a SAS file
is the process of delivering a SAS file from a source host to a target host, either by means of a magnetic medium or across a network. See also Copying a SAS file.

Translation table
is an operating system-specific SAS catalog entry that is used to translate the value of one character to another. Translation tables often are needed to support requirements of National Language Support applications. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. Specify a translation table by using the TRANTAB= system option, the TRANTAB statement in the CPORT and CIMPORT procedures, or the TRANTAB statement in the SAS/CONNECT UPLOAD and DOWNLOAD procedures.

Transport engine
is the facility that transforms a SAS file from its host-specific internal representation to transport format. To create a transport file, explicitly invoke the XPORT engine in the LIBNAME statement when used with the COPY procedure. See also Transport file, Transport format, and Transporting a SAS file.

Transport file
is a sequential file that contains one or more data sets or catalogs or both in transport format.

Transport format
is the internal representation of a transport file. The transport file contains a header, (which describes the content of the file) and the content of the member type (which is represented in binary format). Two distinctive transport formats result from the method that is used to create the transport file. The methods are: 1) the COPY procedure with the XPORT engine or 2) the CPORT and CIMPORT procedures. See also Transport engine, Transport file, and Transporting a SAS file.

Transporting a SAS file
is the process of putting a SAS file in transport format in order to move it between incompatible hosts. The transport process: 1) creates a transport file on the source host, 2) transfers the transport file to the target host, and 3) restores the transport file to native format on the target host. If the source and target hosts run different versions of SAS, the transport process implicitly converts the file only from an earlier SAS version to a later SAS version. See also Architectural compatibility, Converting a SAS file, Transferring a SAS file, Transport file, and Transport format.

Universal header
is attached to the beginning of a SAS file that was created with CEDA. The header contains architectural attributes such as number size, number alignment, data representation, and character encoding. Accessing the universal header, the remote host can determine if the file's format is native or foreign to that of the accessing host. If the file's format is native, then the host can read, write, and update the file. If the file's format is foreign, then the host only can read it. See also Architectural compatibility, CEDA, Foreign host format, and Native host format.

V6 engine
is the default Version 6 engine. This engine accesses SAS files in Version 6 format SAS data libraries.

V7 engine
is the default Version 7 engine. This engine accesses SAS files in Version 7 format SAS data libraries. Version 7 files and Version 8 files are identical.

V8 engine
is the default Version 8 engine. This engine accesses SAS files in Version 8 format SAS data libraries. Version 8 files and Version 7 files are identical.

XPORT
See Transport engine.

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