CHAPTER
15

Preventing and Fixing Problems

Checklist for Transferring and Restoring Transport Files

To avoid potential problems when transferring a transport file to the target host, ensure that these conditions have been met.

1 If transferring across the network, verify that the transport file is transferred in binary format. See “Transferring the Transport File in Binary Format” on page 104 for more information.

2 Verify that the transport file has not been corrupted. See “Verifying That the Transport File Has Not Been Corrupted” on page 104 for more information.

3 Verify that the communications software does not change file attributes. See “Verifying That the Communications Software Has Not Changed File Attributes” on page 104 for more information.

4 Consider invoking the communications software at the target host and getting the transport file from the source host. See “Invoking the Communications Software at the Target Host” on page 105 for more information.

5 Do not mix methods to create the transport file at the source host and then to restore the transport file at the target host. See “Using Compatible Transport Methods at the Source and Target Hosts” on page 105 for more information.

6 Before you transfer a transport file to the target host, validate the integrity of the transport file by restoring it to the source host that created it. See “Validating the Integrity of the Transport File” on page 106 for more information.

7 If transferring by means of tape, use an unlabeled tape. See “Using an Unlabeled Tape” on page 106 for more information.

8 If transferring a large transport file by means of tape, break up the library into multiple libraries and transport each one to tape. See “Dividing a Large Transport File Into Smaller Files for Tape” on page 106 for more information.

Remaining sections explain these topics in detail.

Transferring the Transport File in Binary Format

When transferring a transport file using the communications software, verify that the file is transferred in binary (or image) format. The content of the file must be transferred in sequential bytes without modification.

If you use FTP to move a transport file to the target host, you should first specify BINARY 80 before transferring the file.

If you use PATHWORKS, use the SEQUENTIAL_FIXED attribute when you set the file_server service using PCSA_MANAGER. The default attribute is STREAM, which is not appropriate for moving transport files.

Verifying That the Transport File Has Not Been Corrupted

Verify that your communications software does not insert a carriage return to mark an end of record in the transport file during transfer to the target host. The insertion of carriage returns and line feeds corrupts the transport file, making it impossible to restore at the target host. For details about how to determine this condition, see the recovery actions for “File *libref*.ALL is damaged. I/O processing did not complete” on page 109.

Verifying That the Communications Software Has Not Changed File Attributes

Verify that your communications software does not change file attributes. Here are the required attributes with values:

Logical record length (LRECL)	80
Block size (BLKSIZE)	8000 blocks
Record format (RECFM)	Fixed block

See your communications software documentation for information about controlling these attributes.

At the target host, if you have a transport file that has not been corrupted (carriage returns or line feeds have not been inserted), but whose record block size is incorrect and you are unable to obtain a correctly blocked transport file, you may run a reblocking program to fix the blocks to the correct size. For details, see "Reblocking a Transport File" on page 114.

Invoking the Communications Software at the Target Host

To transfer the transport file to the target host, you may be more successful invoking the communications software at the target host than at the source host. You probably cannot put a file in a location on the target host because you do not have write permission. For example, if transferring a transport file from UNIX to CMS, you are advised to invoke the communications software at the CMS host. Because you probably have read permission at the UNIX host, you can get the transport file and write it to your CMS host.

Using Compatible Transport Methods at the Source and Target Hosts

Do not mix methods to create the transport file at the source host and then restore the transport file at the target host. The methods that you use must be identical or be a companion pair. For example, create and restore a transport file using the XPORT engine and PROC COPY at both the source and target hosts. Likewise, create a transport file using PROC CPORT at the source host and import the transport file using PROC CIMPORT at the target host. *Do not*, 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 restore the transport file at the target host.

To determine the method that was used to create a transport file, use a text editor or an operating system read or view command to read the file on any Version 8 host that represents character data as ASCII.

Note: For information about viewing transport files on hosts that represent character data as EBCDIC, see "Representing EBCDIC as ASCII or Hexadecimal Data" on page 122. △

The XPORT engine creates a file whose first line contains this ASCII text:

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

PROC CPORT creates a file whose first line contains this text:

```
**COMPRESSED** **COMPRESSED** **COMPRESSED**
```

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. △

Validating the Integrity of the Transport File

To validate the integrity of the transport file before it is transferred to the target host, using the appropriate method, try to read it back into native format at the source host.

Here is a PROC COPY example:

```
/* This PROC COPY creates the transport file TRAN. */
libname tran xport 'transport-file';
libname grades 'SAS-data-library';
proc copy in=grades out=tran memtype=data;
run;
/* This PROC COPY reads back transport file TRAN. */
libname grades 'SAS-data-library';
libname tran xport 'transport-file';
proc copy in=tran out=test;
run;
```

Here is a PROC CPORT and PROC CIMPORT example:

```
/* This PROC CPORT creates the transport file. */
libname grades 'SAS-data-library';
filename tran 'transport-file';
proc cport library=grades file=tran;
run;
/* This PROC CIMPORT reads back the transport file. */
filename tran 'transport-file';
libname grades 'SAS-data-library';
proc cimport library=grades infile=tran;
run;
```

For both examples, check the log for error messages.

Using an Unlabeled Tape

When transferring a transport file by means of tape, use an unlabeled tape. Because tape labels are processed differently in different operating environments, reading a file from a standard label tape may be somewhat complicated at the target host.

Dividing a Large Transport File Into Smaller Files for Tape

When transferring a transport file by means of tape, if the transport file exceeds the capacity of one tape, rather than using multi-volume tapes, you should divide the original library into two or more libraries and create a separate, unlabeled tape for each one. The original library can be restored at the target host.

Error and Warning Messages

For all hosts, the most common error and warning messages with recovery actions are presented here. For host-specific messages, see the appropriate operating environment chapter.

Bad Transport File

The primary cause for this message is that you are attempting to use PROC CIMPORT to move a transport file that was created in Version 8 to a host that is running Version 6. You cannot move a transport file from a Version 8 SAS session on a source host to a Version 6 SAS session on a target host.

Another possibility is that either a file was transported in some format other than BINARY or the attributes of the transport file changed while in transit to the target host.

See “Verifying Transfer Format and Transport File Attributes” on page 113 for recovery actions.

An alternative cause is that your site may use a translation table other than the default. A customized translation table is set with the TRANTAB= system option. See *SAS Language Reference: Dictionary* for details about setting the TRANTAB= system option. Verify the value of the TRANTAB= system option:

```
proc options option=trantab;
run;
```

If you discover that your site is using an alternative translation table, you must restore the option to its default value.

```
options trantab=( );
```

Then create the transport file again, transfer it to the target host, and import the file at the target host.

Another possible explanation applies to a source host that runs SAS Release 6.12 and a target host that imports the file at the target host that runs SAS Release, 6.08, 6.09E, or 6.10. Data set sort features (set through the SORTEDBY= data set option) are included in the Release 6.12 CPORT procedure but not in the Release 6.08 CIMPORT procedure.

Use either of two options to recover from this problem:

- Disable the sorting feature by using the SORTINFO= option in the Release 6.12 CPORT procedure. Here is an example:

  ```
  proc cport data=grades.junior
   file='xgrades.junior'
   sortinfo=no;
  ```

- To disable the Release 6.12 sorting feature, use the V608 or V609 engine option in the Release 6.12 CPORT procedure. Here is an example:

  ```
  proc cport data=grades.junior
   file='xgrades.junior' v609;
  ```

The SORTEDBY= data set option information is included in Release 6.12 PROC CPORT.

Catalog file open function is not supported by the XPORT engine

You see this message when you attempt to create a transport file for a catalog or catalog entry by using PROC COPY with the XPORT engine. Instead, you must use PROC CPORT to create a transport file for a catalog or catalog entry and use PROC CIMPORT to import them at the target host.

DATA= or LIBRARY= parameter expected instead of CATALOG=

This message is displayed at the target host when PROC CIMPORT contains a CATALOG= destination member and the source host used PROC CPORT with the LIBRARY= destination member. The target host must use either the DATA= or LIBRARY= member type. Here is an example:

```
proc cport file=in libname=out;
proc cimport infile=in catalog=new;
```

Because the PROC CPORT LIBNAME= option specifies a destination member of type LIBRARY, the PROC CIMPORT must also specify either a LIBNAME= or DATA= option.

In order to select only a catalog entry type from an imported library, specify the ET= option in PROC CIMPORT. To exclude a catalog entry type, use the EET= option. Here are examples:

```
proc cimport infile=in library=new et=program memtype=catalog;
proc cimport infile=in library=new eet=program memtype=catalog;
```

In the first example, only catalog entries of type PROGRAM are imported. In the second example, only catalog entries of type PROGRAM are excluded. MEMTYPE=CATALOG restricts the import to catalogs only.

filename is not a SAS file

You typically see this message when using the CIMPORT procedure to import a data set at the target host. There are two possible explanations.

The transport file that you are trying to import by using PROC CIMPORT may have been created by using the XPORT engine with either the COPY procedure or the DATA step. Read the beginning of the file to determine how the transport file was created. If the XPORT engine created the transport file, the beginning of the file contains this text:

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

If the CPORT procedure created the transport file, the beginning of that file contains this text:

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

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

If incompatible methods were used to create and then restore the transport file, then use the correct method to restore the transport file.

Another possible explanation is that your site may use a translation table other than the default. For recovery actions for this problem, see “Bad Transport File” on page 107.

Entry type *catalog-entry-type* is not supported by CPORT

Transporting this catalog entry type between hosts and across SAS releases is not supported.

Because you cannot retrieve the definitions from the module itself, you can try to move the SAS statements that defined the entry type (such as IML modules) to the target host and then re-create the modules.

Entry type *catalog-entry-type* is not compatible to earlier release

You see this message when you attempt to use PROC CPORT to move a catalog entry from Version 8 back to Version 6. Version 8 does not support the backward compatibility of this catalog entry.

File *library.member*.DATA has too long a member name for the XPORT engine

This message appears when you use the XPORT engine with PROC COPY to move a data set on a Version 8 source host whose name exceeds eight characters to a Version 6 library. Here is an explicit example of such a message:

```
ERROR: The file OUT.THIS_IS_LONG_NAMED_DATA.DATA
  has too long a member name for the XPORT engine.
```

The member name **THIS_IS_LONG_NAMED_DATA** exceeds the eight-character member name length, which is enforced by the Version 5 feature set in which the XPORT engine was introduced.

The VALIDVARNAME SAS system option and the assigned value of V6, which enables automatic truncation of long variable names, does not support member names. To recover, copy the member to another member whose name does not exceed 8 characters and retry the transport operation.

File *library.member*.DATA has too long a member name for the V6 engine

This message appears when using PROC COPY to move a data set on a Version 8 source host whose name exceeds 8 characters to a Version 6 library. Here is an explicit example of such a message:

```
ERROR: The file V6LIBMYDATABASE.DATA
  has too long a member name for the V6 engine.
```

The Version 8 data set name **MYDATABASE** exceeds the maximum member name length of 8 that is supported in Version 6. Version 6 interprets the data set name **MYDATABASE** as containing 10 characters, which exceeds its maximum length of 8.

The VALIDVARNAME SAS system option and the assigned value of V6, which enables automatic truncation of long variable names, does not support member names. To recover, rename the member or copy it to another member whose name does not exceed 8 characters and retry the transport operation.

File *libref*.ALL is damaged. I/O processing did not complete

This message typically alerts you to a file corruption. The likely explanation is that your site's communications software inserted carriage returns into the transport file.

At the target host, you can use a host-specific utility (such as the UNIX hexadecimal dump utility **xd**) to view the transport file in hexadecimal format to determine if carriage returns were inserted. See the UNIX **xd**(1) manual page for details. As another example, for OS/390, use the SPF 1 command for browsing, select a data set, and enter **hex on** in the command line.

Example Code 15.1 on page 110 shows an example of a transport file that contains a carriage return character (0D) and a line feed character (OA) toward the end of the first record. See the 0D and 0A hex values in the first two positions of the last line.

Example Code 15.1 Hexadecimal Representation of a Transport File

```
48 45 41 44 45 52 20 52 45 43 4F 52 44 2A 2A 2A HEADER R ECORD***
2A 2A 2A 2A 4C 49 42 52 41 52 59 20 48 45 41 44 ****LIBR ARY HEAD
45 52 20 52 45 43 4F 52 44 21 21 21 21 21 21 21 ER RECOR D!!!!!
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 00000000 000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 20 20 00000000 0000
0D 0A 53 41 53 20 20 20 20 20 53 41 53 20 20 20 ...SAS    SAS
```

If you do not see carriage return or line feed characters, another form of corruption that is not immediately apparent may have occurred. To test this possibility, at the target host, create another transport file from a member of the same type and then view its hexadecimal representation. Compare the appearance of the assumed uncorrupted file that you just created with the suspected corrupted file that you are trying to restore. A visual comparison may prove that the transport file that you are trying to restore is indeed corrupt. In this case, re-create the transport file at the source host, transfer it, and restore it at the target host.

At the source host, determine whether the transport file's attributes include carriage returns. For information about listing and correcting file attributes, see the appropriate operating environment chapter.

At the source host, transfer the transport file to the target host again.

If you are still unable to restore a transport file that has the correct file attributes, you may try using the re-blocking program in "Reblocking a Transport File" on page 114.

Given transport file is bad

See "Bad Transport File" on page 107 for recovery actions.

Internal error from getting data

This message typically appears because either a file was transported in some format other than BINARY or the attributes of the transport file changed while in transit to the target host.

See "Verifying Transfer Format and Transport File Attributes" on page 113 for recovery actions.

Invalid data length

This message typically appears because either a file was transported in some format other than BINARY or the attributes of the transport file changed while in transit to the target host.

See "Verifying Transfer Format and Transport File Attributes" on page 113 for recovery actions.

Member or library unavailable for use in file *file*

This message typically appears because either a file was transported in some format other than BINARY or the attributes of the transport file changed while in transit to the target host.

See “Verifying Transfer Format and Transport File Attributes” on page 113 for recovery actions.

Another possible explanation applies to a Release 6.12 SAS session on a source host and a Release 6.08 SAS session on a target host. Data set sort features (specified by using the SORTEDBY= data set option) are included in the Release 6.12 CPORT procedure but not in the Release 6.08 CIMPORT procedure.

Use either of two options to recover from this problem:

- Disable the sorting feature by using the SORTINFO= option in the Release 6.12 CPORT procedure. Here is an example:

  ```
  proc cport data=grades.jr file='tranfile.jr' sortinfo=no;
  ```

- To disable the Release 6.12 sorting feature, use the V608 engine explicitly in the Release 6.12 CPORT procedure. Here is an example:

  ```
  proc cport data=grades.jr file='tranfile.j v608;
  ```

The SORTEDBY= data set option information is included in Release 6.12 PROC CPORT.

More library members exist in the input file. For all of them to get converted, please specify LIBRARY=libref parameter in the PROC statement

This warning message is displayed at the target host when PROC CIMPORT contains a DATA= destination member and the source host used PROC CPORT with the LIBRARY= destination member. Although, the target host successfully imports only one data set, the message indicates that other members are contained in the library that can also be imported. Here is an example:

```
proc cport file=in library=out;
proc cimport infile=in data=new;
```

In order to expand the import operation to include the entire contents of destination library, specify the LIBRARY= option rather than the DATA= option in PROC CIMPORT.

PROC SQL will not store a V8 view into a V6 library

This message is typically displayed when you use the XPORT engine to create a Version 8 PROC SQL view in transport format in a Version 6 library. However, you can use the XPORT engine to create an SQL table.

To recover, transport the data set that contains the SQL table to the target host and re-create the PROC SQL view there.

Requested function is not supported

This message indicates a failure to move a library from a Version 8 source host to a library on a Version 6 target host because of cross-version incompatibilities. For example, Version 8 features such as generations data sets and integrity constraints are not supported.

To recover, you must remove Version 8 features from the library or the member to be moved to the library on the Version 6 host and retry the transport operation. Preceding notes in the log can hint at the offending Version 8 feature that is not supported. Here is an example:

```
NOTE: Integrity constraint mc defined.
```

You can infer from this message that Version 6 does not support integrity constraints.

For tips on how to remove Version 8 features, see the recovery actions for these messages: "File *library.member*.DATA has too long a member name for the V6 engine" on page 109 and "Variable name *XXXXXXXXX* is illegal for file *Version-6-data-set* "on page 112.

Truncated record

This message typically appears because either a file was transported in some format other than BINARY or the attributes of the transport file changed in transit to the target host.

See "Verifying Transfer Format and Transport File Attributes" on page 113 for recovery actions.

This message can indicate that the transport file was moved to a virtual disk or shared disk with other operating environments such as DOS, Macintosh, or UNIX. For recovery actions, see the appropriate operating environment chapter.

Updating not allowed for *libref.member-name* because it was created for a different operating system

This message appears when a host attempts to update a file whose format is foreign to that of the accessing host. Use PROC CONTENTS on the file to verify the file's data representation. A data represention of FOREIGN proves that the formats of the file and the accessing host are incompatible.

UTILITY FILE OPEN function is not supported by the XPORT engine

This message appears when you attempt to use PROC COPY with the XPORT engine to create a transport file for a utility file, such as an MDDB. The XPORT engine does not support utility files.

Variable name *XXXXXXXXX* is illegal for file *Version-6-data-set*

This message appears when using PROC CIMPORT to move a Version 8 data set that contains long variable names to a Version 6 data set. Here is an explicit example of such a message:

```
ERROR: The variable name Region_Of_The_Country
is illegal for file V6LIB.CITY.DATA.
```

The Version 8 variable name **Region_Of_The_Country** exceeds the maximum variable name length of 8 that is supported in Version 6. To recover, in the SAS session on the local host, set the VALIDVARNAME SAS system option to V6 to enable automatic truncation of long variable names. Retry the transport operation. Here is an example of setting this variable:

```
options validvarname=v6;
```

In this example, **Region_Of_The_Country** truncates to **Region_O** However, if the data set contains multiple variables whose first 8 characters conflict, Version 8 uses a truncation algorithm that ensures uniquely truncated variable names. For details, see "Regressing SAS Data Sets from Version 8 to Version 6 Format" on page 19.

Verifying Transfer Format and Transport File Attributes

Verify that the communications software that you use to transfer the transport file is in BINARY format. If you use FTP, for example, you would explicitly enter the FTP BINARY command. Here is a sample invocation of FTP:

```
ftp
> open host
> binary
> get file file
> close
> quit
```

For details about FTP, see Chapter 4, "Transferring a Transport File or a CEDA File," on page 31.

Even if your communications software claims to submit transport files in an appropriate format by default, *always* be certain of binary format by explicitly specifying it. For details about how to specify the transfer format, consult your communications software documentation.

Also, verify the file attributes of the transport file, which are required in order to restore the file at the target host. Although some target hosts may not need file attributes, the transfer method (tape and network) always does. See "Transport File Attributes" on page 31 for a list of hosts that require file attributes. Problems can result when the file attributes that are required by the target host and those applied by the transfer method conflict.

Verify file attributes that are required by the target host. How you list and set file attributes varies by host. See the appropriate host environment chapter for this information.

Also verify the file attributes that the transfer method sets. For example, if using FTP, you set file attributes in an FTP command. Here is a sample invocation of FTP:

```
ftp
> open host
> binary
> locsite recfm=fb blocksize=8000 lrecl=80
> get file file
> close
> quit
```

If transferring a transport file across a network, consult your communications software documentation. For information about transferring a file by means of tape, see the appropriate host environment chapter.

If you can correct the problem, re-create the transport file at the source host, transfer it to the target, and restore it again.

If the problem persists, try to reblock the transport file and retry transporting it. See “Reblocking a Transport File” on page 114.

Reblocking a Transport File

At the target host, if you determine that the transport file has an incorrect block size and you are unable to obtain a transport file that contains the correct block size, then use the reblocking program to reblock the transport file.

Note: The transport file against which the reblocking program is run must be uncorrupted; that is, no extra carriage returns or line feeds can be inserted. If the transport file is known to be corrupted, the reblocking program will fail. △

This program copies the transport file and produces a new transport file that contains 80-byte fixed block records.

Example Code 15.2 Program that Reblocks a Transport File

```
data _null_;

  /* Note: the INFILE and FILE statements must */
  /* be modified. Substitute your file names.  */
  infile 'your_transport.dat' eof=wrapup;
  file 'new_transport.dat' recfm=f lrecl=80;

  length irec $16 outrec $80 nullrec $80;
  retain count 1 outrec nullrec;
  input inrec $char16. @@;
  substr(outrec, count, 16) = inrec;
  count + 16;
  if (count > 80) then do;
    put outrec $char80.;
    count=1;
  end;
  return;

wrapup:;
  file log;
  nullrec = repeat('00'x,80);
  if outrec = nullrec then do;
    put ' WARNING: Null characters may have been'
        ' added at the end of transport file by'
        ' communications software or by a copy'
        ' utility. For a data set transport file,'
        ' this could result in extra null'
        ' observations being added at the end'
        ' of the last data set.';
   end;
run;
```

In the example, the record format of the original transport file is fixed and the record length is evenly divisible by 16.

If your record type is fixed, but the record length is *not* evenly divisible by 16, then determine the greatest common denominator that is divisible by both 80 and the transport file record length. Substitute this number for all occurrences of 16 in the preceding program.

For example, 80 is evenly divisible by 1, 2, 5, 8, and 10. A fixed record length of 99 for a transport file is evenly divisible by 1, 3, 9, and 11. Thus, 1 is the only common denominator, and is, therefore, both the lowest common denominator and the greatest common denominator.

Note: If the transport file has a variable length record type, then use 1 instead of 16 as the greatest common denominator. △

CAUTION:

For a transport file that contains data sets, some communications software pads the final record with null characters. The reblocking program may add extra observations that contain all zero values to the end of the final data set in a library. △

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

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-480-2

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, by any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st printing, October 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

IBM®, AIX®, DB2®, OS/2®, OS/390®, and System/390® are registered trademarks or trademarks of International Business Machines Corporation. ORACLE® is a registered trademark or trademark of Oracle Corporation. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

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