

# How to...

04/24/96

## Read Compressed SAS Transport Files Directly in your SAS Program

SAS users who are short on disk space on UNIX often compress their large data sets. Depending on how the data are stored - as raw data, a permanent SAS data set, or a SAS transport file; SAS may or may not be able to read the UNIX compressed data directly. Compressed raw data can be read directly and this is documented in SSC Pub. #7-4, *Using SAS on UNIX*.

Often, it is more convenient to work with permanent SAS data sets than it is to work with raw data. Unfortunately, though, *UNIX* compressed permanent SAS data sets cannot be read directly by SAS. SAS compressed data sets can be read directly by SAS. (Do not confuse UNIX compression with the SAS option COMPRESS=YES; they are two different things.) They must first be uncompressed. And, uncompressing the data defeats part of the purpose of having them compressed in the first place. It turns out though, that if you convert your SAS data set into a SAS transport file and compress that file, then SAS can read the file directly.

A transport file is very similar to its SAS data set counterpart. It offers most of the advantages of a permanent SAS data set plus it is operating system independent. For a complete discussion of permanent SAS data sets and SAS transport files including how to create them, refer to SSC Pub. #7-4, *Using SAS on UNIX*.

This handout describes how to read a compressed SAS transport file in a five-step process:

- 1) Create a named pipe.
- 2) Write the compressed transport file to the pipe.
- 3) Issue a SAS libname statement to identify the pipe to the XPORT engine.
- 4) Use PROC COPY or a DATA step to read in the SAS transport file.
- 5) Remove the pipe.

If you are unfamiliar with named pipes, they are described in SSC Pub. #7-5, *Using Compressed Files*. This handout assumes you are familiar with this UNIX concept.

### Creating a Named Pipe

A FILENAME statement along with the PIPE device type allows you to issue UNIX commands within your SAS program. An X command can also be used for this purpose but causes certain system problems when used for reading compressed SAS transport files so is no longer recommended.

The FILENAME statement below is used to issue the UNIX `mknod` command which creates named pipes:

```
filename MKPIPE pipe 'mknod piper p';
data _null_;
    infile MKPIPE;
run;
```

MKPIPE is a fileref in the FILENAME statement above. The INFILE statement in the DATA step which follows references this fileref which causes the `mknod` command to be executed. `piper` is the name assigned the named pipe.

### Writing the File to the Named Pipe:

Next, a FILENAME statement is used to write to the pipe in a similar fashion as was used in the first step:

```
filename NPIPE pipe 'zcat heart.xpt.Z >
piper &';
data _null_;
    infile NPIPE;
run;
```

The `zcat` command writes the data to the named pipe and runs in the background.

## Issuing a SAS Libname Statement to Identify the Pipe

Next, you need to issue a SAS LIBNAME statement specifying the XPORT engine to identify the pipe. The name you specify for the file will be the named pipe, not the transport file itself:

```
libname READ xport 'piper';
```

Recall that READ in the statement above is the libref. For more information on librefs and the LIBNAME statement, refer to SSC Pub. #7-4, *Using SAS on UNIX*.

## Using PROC COPY or a DATA Step to Read the Transport File

If your data set requires no DATA step modifications, you can use PROC COPY to convert the transport file to a temporary SAS data set before using it with SAS procedures:

```
proc copy in=READ out=WORK;
  select person;
run;
```

The IN= refers to the libref specified in a prior LIBNAME statement identifying the transport file to be read (READ, for this example). The OUT= tells SAS where you want the SAS data set written. In this example WORK is used because WORK is a special libref reserved for writing temporary SAS data sets. The SELECT statement is where you specify the name of the original permanent SAS data set (the name you specified on the right side of the period of the DATA statement used to create the permanent SAS data set in a prior program.) Hint: Do not confuse the name of the SAS transport file with the name of the SAS data set on this SELECT statement. The two names may be different. For this example, the SAS data set name is the same as the filename (not including the file extension) of the transport file, but it need not have been.

If your data set requires some DATA step

modifications prior to analysis, you can use this DATA step to read in the transport file as well as to carry out your modifications instead of a PROC COPY step followed by a DATA step which would be much less efficient:

```
data modify;
  set read.person;
  modifications...;
run;
```

Note that this example creates a temporary SAS data set. A permanent SAS data set could have been created instead.

## Removing the Pipe

It is good practice to remove the pipe before you end your SAS program. Otherwise, the pipe will remain on your disk. The SAS statements below will remove the pipe:

```
filename RMNPIPE pipe 'rm piper';
data _null_;
  infile RMNPIPE;
run;
```

Following is a complete SAS program for the example described above:

```
/* Create a pipe */
filename MKPIPE pipe 'mknod piper p';
data _null_;
  infile MKPIPE;
run;

/* Write the transport file to write to
the pipe */
filename NPIPE pipe 'zcat heart.xpt.Z >
piper &';
data _null_;
  infile NPIPE;
run;

/* Import the SAS data set */
libname READ xport 'piper';
proc copy in=READ out=WORK;
run;

proc print;
run;

/* Remove the PIPE */
filename RMNPIPE pipe 'rm piper';
data _null_;
```

```
infile RMNPIPE;  
run;
```

