Stat/Transfer

for

UNIX

Circle Systems

1001 Fourth Avenue, Suite 3200, Seattle, WA 98154
(206) 682-3783 FAX (206) 328-4788
stat-transfer@circlesys.com
http://www.stattransfer.com

STAT/TRANSFER
PERSONAL LICENSE AGREEMENT

This is a legal agreement between you (either an individual or entity), the end user, and Circle Systems, Inc. If you do not agree to the terms of this agreement, promptly return the disk package and the accompanying items (including written materials) to the place you obtained them for a full refund.

GRANT OF LICENSE. Circle Systems grants you the right to use one copy of the Stat/Transfer program (the "SOFTWARE") on a single computer ("Dedicated Computer"). You may transfer the SOFTWARE to another single computer PROVIDED you do so no more often than once every thirty (30) days and no copies of the SOFTWARE licensed herein are retained for use on any other computer. You may also use the SOFTWARE on a portable or home computer.

For the purposes of this section, "use" means loading the SOFTWARE into RAM, as well as installation on a hard disk or other storage device. You may access the SOFTWARE from a hard disk, over a network, or any other method you choose, so long as you otherwise comply with this license agreement.

COPYRIGHT. The SOFTWARE is owned by Circle Systems and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the software like any other copyrighted material (e.g. a book or musical recording) ***except*** that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk, provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the software.

OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE, but you may transfer your rights under this agreement on a permanent basis, provided you transfer the SOFTWARE and all accompanying written materials, retain no copies, and the recipient agrees to the terms of this agreement. You may not reverse engineer, decompile or disassemble the SOFTWARE. If the SOFTWARE is an update, any transfer must include the update and all prior versions.

LIMITED WARRANTY

LIMITED WARRANTY. Circle Systems warrants that the software will substantially conform to the published specifications and to the documentation. Circle Systems also warrants that the magnetic media on which the software is distributed and the documentation are free from defects in materials or workmanship.

CUSTOMER REMEDIES. Circle systems will replace defective media or documentation or correct substantial software errors at no charge, provided you return the items with dated proof of payment to Circle Systems within 90 days of the date of delivery. If Circle Systems is unable to replace defective media or documentation or correct substantial software errors, Circle Systems will refund the license fee. These are your sole remedies for any breach of warranty.

NO OTHER WARRANTIES. Except as provided above, Circle Systems makes no warranty or representation with respect to the SOFTWARE and accompanying written materials, either express or implied, including but not limited to their quality, performance, merchantability or fitness for a particular purpose.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. Because software is inherently complex and may not be completely free from errors, you are advised to verify your work. In no event will Circle Systems be liable for direct, indirect, incidental or consequential damages arising out the use or inability to use the SOFTWARE and its accompanying written materials, even if advised of the possibility of such damages. In particular, Circle Systems is not responsible for any costs including, but not limited to, those incurred as a result of lost profits or revenue, loss of use of the software, loss of data, the cost of recovering such data, the costs of substitute software, claims by third parties or similar costs. In no case shall Circle Systems' liability exceed the amount of the license fee.

The warranty and remedies set forth above are exclusive and in lieu of all others, oral or written, express or implied. No Circle Systems dealer, distributor, agent or employee is authorized to make any modifications or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or limitations of liability for incidental or consequential damages, so the above limitations or exclusion may not apply to you.

Table of Contents

Introduction

Stat/Transfer is designed to simplify the transfer of statistical data between different programs.

Data generated by one program is often needed in another context, either for analysis, for cleaning and correction, or for presentation. However, not only must the data be transferred, but in addition, the variables generally must be re-described for each program with additional information, such as variable names, missing values and value and variable labels. This process is not only time-consuming, it is error-prone. For those in possession of data sets with many variables, it represents a serious impediment to the use of more than one program.

Stat/Transfer removes this barrier by providing an extremely fast, reliable and automatic way to move data. Stat/Transfer will automatically read statistical data in the internal format of one of the supported programs and will then transfer as much of the information as is present and appropriate to the internal format of another.

Stat/Transfer preserves all of the precision in your data by storing it internally in double precision format. However, on output, it will, where possible, automatically minimize the size of your output data set by intelligently choosing data storage types that are only as large as necessary to preserve the input precision.

In addition to converting the formats of variables, Stat/Transfer also processes missing values automatically.

Stat/Transfer can save hours and even days of manual labor, while at the same time eliminating error. Furthermore, you gain this speed and accuracy without losing flexibility, since Stat/Transfer allows you to select just the variables and cases you want to transfer.

Differences between the Unix and Windows Versions

The Stat/Transfer command processor on Windows and on Unix platforms share a common code base. However, there are a few differences in “style” between the two versions and also in the file types supported. If you are familiar with the Windows version of Stat/Transfer, you should read the following sections.

Differences in Supported File Types

The following formats are not yet available on Unix:

ODBC
Access
Paradox
Minitab
Statistica

Style Differences

Switches

The switch character used with Unix commands is the dash, rather than the slash used with DOS commands.

For example, to copy a file with optimizing and no warning prompt when overwriting an existing file, type:

```
copy indata.ssd01  out.dta   -o  -y
```

Alternative Commands

There are several changes to the built-in operating system commands under Unix:

‘cp’ is equivalent to ‘copy’
‘ls’ is equivalent to ‘dir’
‘!’ or ‘system’ takes the place of ‘DOS’

Case

Stat/Transfer commands are not case sensitive, but Unix file systems are. Therefore, file names and paths must always be in the correct case, but case does not matter with commands when using Stat/Transfer for Unix.

Messages

When run from the operating system prompt, Stat/Transfer for Unix by default will issue messages only if errors occur. If you wish more verbose output, use the ‘-v’ switch.

Wildcards

When using the Unix version, if you are running a wildcard transfer from the operating system prompt, the arguments must be enclosed in quotes to prevent the shell from expanding the wildcards before Stat/Transfer sees them.

For example:

```
st   "in/*.sd2" "out/*.dta"
```

Supported File Types

Version 5.0 of Stat/Transfer supports the following file types:

- 1-2-3
- ASCII - Delimited
- dBASE and compatible formats
- Epi Info
- Excel
- FoxPro
- Gauss

- HTML tables
- JMP
- LIMDEP
- Matlab
- Mineset
- OSIRIS (read-only)
- Quattro Pro
- SAS for Windows and OS/2
- SAS for Unix
- SAS Transport
- S-PLUS
- SPSS Data
- SPSS Portable
- Stata
- SYSTAT

See the sections on specific file types for more information.

Transfer Between Worksheet Files

Note that Stat/Transfer does not support conversions from one type of worksheet to another, such as a Lotus 1-2-3 worksheet to an Excel worksheet. These types of conversions are not supported since it is usually possible to do them within your worksheet program.

Because we make every effort to keep up with changes in the file formats of popular software, be sure to check our web site for the latest information on which versions of these programs are supported.

You can also get current information about Stat/Transfer by visiting our web site at **http://www.stattransfer.com**.

Installation

Installing Stat/Transfer

Stat/Transfer for Unix runs on the following platforms:

DEC Alpha (OSF/1)
HP-9000 (HP-UX)
IBM RS/6000 (AIX)
Intel Pentium (Linux)
SGI MIPS (Irix)
Sun SPARC (Solaris)

The Stat/Transfer executable file for each of these platforms is available on our web site as a single, uncompressed file, named ***st***:

DEC Alpha (OSF/1)	ftp://ftp.stattransfer.com/alpha/st
HP-9000 (HP-UX)	ftp://ftp.stattransfer.com/hp/st
IBM RS/6000 (AIX)	ftp://ftp.stattransfer.com/aix/st
Intel Pentium (Linux)	ftp://ftp.stattransfer.com/linux/st
SGI MIPS (Irix)	ftp://ftp.stattransfer.com/sgi/st
Sun SPARC (Solaris)	ftp://ftp.stattransfer.com/solaris/st

Download the appropriate ***st*** file and save it to any convenient location.

Trial Mode Software

The downloaded file will be in "trial mode." Your software will work as if it were a full version, except that one out of approximately sixteen cases will be deleted from your output file. A message will warn you of this.

If you choose to purchase the software, we will send you a licensing file by e-mail that will enable it.

Licensing File

A licensing file must be present for the program to run correctly. The license is a small text file, which is generally sent as an e-mail attachment. If the information in this file is altered or the file is missing, the program will revert to trial mode and one out every sixteen cases will be deleted.

If you have purchased a personal license, the license file must be installed in your home directory and it must be named ***.circlesys.lic*** (note the leading dot).

If you have purchased a site license, the license file should be named ***circlesys.lic*** (no leading dot). It should be installed in ***/usr/lib/circlesys/*** or, alternatively, it should have a name and location pointed to by the environment variable STLIC. For example, if you wanted to put the license file in ***/usr/lib/lic*** and name it ***st.lic***, you could set the following environment variable:

```
STLIC=/usr/lib/lic/st.lic
```

The license can also be located in the same directory as the program file ***st***, as long as the directory is on your path.

If you find that you are having trouble installing your license file, type

```
st license
```

and Stat/Transfer will type a log of its activity as it searches for your license.

Using Stat/Transfer

Stat/Transfer for Unix will perform transfers using a copy-like syntax, which allows "batch" jobs to be run for repetitive transfers. You can select variables and cases, automatically optimize the output variable types, or change output types manually.

When running Stat/Transfer, you can give commands in either of two ways: directly, or from a stored command file which is executed at the Stat/Transfer or operating system prompt.

Calling Stat/Transfer

At the operating system prompt, type:

```
st
```

In order for the operating system to find Stat/Transfer, you will need to be in the directory in which you installed Stat/Transfer, or have that directory on your path, or specify the path to Stat/Transfer when you call it (for example, **~/bin/st**).

Transfers from within Stat/Transfer

If no other parameters are entered, you will see the Stat/Transfer prompt. You can then enter commands or execute a command file.

Transfers from the Operating System Prompt

If you give input and output file names or a command file name after the 'st' command, as discussed on Page 8, you will carry out an immediate data transfer, without seeing the Stat/Transfer prompt.

The Copy Command

The basic command used when running Stat/Transfer is COPY,

COPY *infilename.ex1 outfilename.ex2 -x1 -x2 ...*

If Stat/Transfer is being used from the operating system prompt, then the COPY command is implicit

ST *infilename.ex1 outfilename.ex2 -x1 -x2 ...*

Details are given below.

Running a Transfer from within Stat/Transfer

The COPY command at the Stat/Transfer prompt is used in the following way:

COPY *infilename.ex1 outfilename.ex2 -x1 -x2 ...*

where *infilename.ex1* and *outfilename.ex2* are the input and output files, respectively and *.ex1 a*nd *.ex2*, are standard extensions used to determine the type of file that is read and written. The file names can be complete file specifications.

Stat/Transfer commands are not case sensitive, but Unix file systems are. Therefore, file names and paths must always be in the correct case, but case does not matter with commands.

Parameters *-xi* following the file names allow some options to be selected, such as automatic optimization of variables or suppression of warning messages.

By default, all of the variables and cases will be transferred.

For example, to copy all of the variables and all of the cases from an Excel file, ***indata.xls***, to a SAS file, ***outdata.sd2***, type:

```
copy  indata.xls  outdata.sd2
```

Stat/Transfer determines the file type by looking for standard extensions. An alternative for specifying the file type is to use a “file-type tag” instead of a standard extension. These are discussed on Page 9.

You can give additional commands before you give the COPY command at the Stat/Transfer prompt. These allow you to select cases and variables, to manually change output variable types and to set a number of options.

Note that if you carry out a transfer interactively from the operating system prompt, the additional commands will not be available, except for SET commands that may be given in a startup options file (see Page 24.)

Wildcard Transfers

Wildcards can be used to copy multiple files in one transfer. For example:

```
copy    \in\*.dta    \out\*.sd2
```

will convert all of the Stata files in the directory ***\in*** to SAS files in the directory ***\out***.

Standard wildcards work in the input file ('?' and '*'). The output filenames should always be specified with an asterisk and will be consist of the input filename, with the new extension.

We recommend that you use wildcard specifications with an input directory that contains just the files you wish to transfer. To avoid the possibility of overwriting existing files or of being prompted about this possibility, the directory should be empty on the output side. You may also specify the '-Y' parameter, discussed on Page 9, to suppress prompting, but you should do so with extreme caution.

Note that when you are running a wildcard transfer from the operating system prompt, as discussed below, the arguments must be enclosed in quotes.

Running a Transfer from the Operating System Prompt

You can run a transfer from the operating system prompt by typing the following:

st *infilename.ex1* *outfilename.ex2* *-x1* *-x2* ...

where *infilename.ex1* and *outfilename.ex2* are the input and output files, respectively, just as they are for the COPY command used within Stat/Transfer. File-type tags are used in the same way, as well. The same parameters, *-xi*, can be used from Stat/Transfer prompt and the operating system prompt.

When a transfer is run interactively from the operating system prompt, the options set with SET commands, described on Pages 24 - 36, are available only if they are included in a startup options file (see Page 24.)

When a transfer is run interactively, the commands that control variable selection and case selection are not available. Thus, all variables and all cases will be transferred. Carrying out a transfer interactively from the operating system prompt is equivalent to copying the complete file from one format to another.

For example, to transfer all of the variables and cases from an Excel file, ***indata.xls*** to a Stata file, ***outdata.dta***, type:

```
st  indata.xls   outdata.dta
```

In order to run a transfer from the operating system prompt line, you will need to be in the directory in which you installed Stat/Transfer, or have a path defined to it, or specify the path to Stat/Transfer when you call it.

Wildcard Transfers

Wildcard transfers can be specified from the operating system prompt, just as they are from the Stat/Transfer prompt. However, when you are running a wildcard transfer from the operating system prompt, the arguments must be enclosed in quotes to prevent the shell from expanding the wildcards before Stat/Transfer sees them. For example,

```
st  "\in\*.dta"  "\out\*.sd2"
```

Specifying the File Types

When a COPY command is given, Stat/Transfer determines the file types of both the input and output files by looking for standard extensions.

An alternative for specifying the file type is to use a “file-type tag” instead of a standard extension.

Standard Extensions

If you use standard file extensions, these will tell Stat/Transfer what kind of file is to be read or written.

Standard File Extensions

1-2-3	WK*
ASCII - Delimited	TXT, CSV
dBASE and compatibles	DBF
Epi Info	REC
EXCEL	XLS
FoxPro	DBF
Gauss	DAT
HTML	HTM
JMP	JMP
LIMDEP	CPJ
Matlab	MAT
Mineset	SCHEMA
OSIRIS	DICT
Quattro Pro	WQ*, WB*
SAS for Windows and OS/2	SD2, SAS7BDAT
SAS for Unix	SSD*, SAS7BDAT
SAS Transport Files	XPT
S-PLUS	[none]
SPSS Data Files	SAV
SPSS Portable Files	POR
Stata	DTA
SYSTAT	SYS

A list of standard extensions can be obtained by typing:

```
help formats
```

at the Stat/Transfer prompt.

File-Type Tags

If your file does not have a standard extension, you can precede the file name by a file-type tag, which indicates the file type.

For example, if you want to write a Windows SAS file that has a ***.dat*** extension instead of an ***.sd2*** extension, you can type

```
copy indata.xls sas outdata.dat
```

where 'sas' indicates that the output file is to be a SAS Windows file.

The file-type tags used by Stat/Transfer are:

File-Type Tags

1-2-3	123
ASCII - Delimited	delim
dBASE and compatibles	xbase
Epi Info	epi
EXCEL	excel
FoxPro	xbase
Gauss	gauss
HTML	html
JMP	jmp
LIMDEP	limdep
Matlab	matlab
Mineset	mineset
OSIRIS	osiris
Quattro Pro	quattro
SAS V6 for Windows and OS/2	sas
SAS V7 and V8 for Windows	sas-win
SAS V6 for Unix - HP, Sun, IBM	sas1
SAS V6 for Unix - DEC	sas4
SAS V7 and V8 for Unix	sas-sun
SAS Transport Files	sasx
S-PLUS for Windows & OS/2 &DEC Unix	splus
S-PLUS for Unix - HP, Sun, IBM	splus-hl
SPSS Data for Window`s & OS/2 &DEC Unix	spss
SPSS Data for Unix - HP, Sun, IBM	spss-hl
SPSS Portable	spssp
Stata	stata
SYSTAT	systat

When you are using Stat/Transfer, a list of tags can be obtained by typing:

```
help formats
```

Distinguishing Between File Names and File-type Tags

Stat/Transfer uses the presence of a period '.' to differentiate between extensions and file-type tags. Therefore, when using the COPY command, you must either use a period in your file name or, if the name does not contain a period, then you must use one after the file name.

For example, if you have a Stata file named ***populat***, and you are going to a 1-2-3 file, ***populat.wk3,*** the COPY command would look like:

```
copy stata populat. populate.wk3
```

The file-type tag must be used since the input file does not have a standard extension. The period after the file name tells Stat/Transfer that this is the file name.

Special Cases when Specifying Files

Specifying S-PLUS Files

A period must always be added to standard S-PLUS file names, since these do not have any extension. No file-type tag is needed, however, since the standard extension for S-PLUS files is ‘none’.

For example, to transfer the S-PLUS file ***indata*** to a Matlab file ***outdata.mat***, type

```
copy indata. outdata.mat
```

The period tells Stat/Transfer that ***indata*** is a file name, not a file-type tag.

Selecting Worksheet Pages

Whenever you select a worksheet as input, Stat/Transfer will check to see if multiple pages are present. If more than one page is found, the first page will be used as the input data set unless you select another one.

If the data you wish to use are on a different page, you must use the parameter ‘-T*pagenumber*’ where *pagenumber* is the page you wish to select.

COPY *inworksheetname.ex1* *outfilename.ex2* -t*pagenumber*

For example, to transfer the data from the third sheet of the Excel worksheet ***inwork.xls*** to the Stata file ***outfile.dta***, type:

```
copy inwork.xls outfile.dta -t3
```

Selecting Members of SAS Transport Files

Whenever you select a SAS Transport file as input, the first member will be the one used as the input data set, unless you select another one.

If the data you wish to use in a different member, you must use the parameter ‘-T*membername*’, where *membername* is the member you wish to select.

COPY *infile.xpt* *outfilename.ex2* -t*membername*

For example, to transfer the data from the member ***part4*** of the SAS Transport file ***indata.xpt*** to the Gauss file ***outdata.dat***, type:

```
copy indata.xpt outdata.dat -tpart4
```

Overwriting Output Files

By default, Stat/Transfer will check to see if the destination file already exists and warn you that an existing file is about to be overwritten. You can suppress this warning using the parameter ‘-Y’.

Options Set by Parameters after the COPY Command

Several options are set with parameters of the form '-*xi*'. These are given after the COPY command:

```
COPY  infilename.ex1   outfilename.ex2   -x1  -x2 ...
```

and can be given in any order. They can also be used at the operating system prompt:

```
ST  infilename.ex1   outfilename.ex2   -x1  -x2 ...
```

Options for Input Data Sets

The parameter '-T*pagenumber*', used for selecting worksheet pages, and the parameter '-T*membername*', used for selecting members of SAS transport files are discussed on the previous page.

Options for Variables

Automatic Optimizing of Output Variable Type

Stat/Transfer attempts to produce the smallest possible output data set. By default, information from the data file dictionary will be used to determine the output variable types.

As discussed on Page 20, you can choose to have Stat/Transfer make an additional optimization pass through your data to determine more information about each variable. In order to tell Stat/Transfer to automatically optimize the output variable types, the optimization parameter '-O' is used:

```
COPY  infilename.ex1   outfilename.ex2   -o
```

When this parameter is given, Stat/Transfer will make an optimization pass when the output file type will benefit from optimization.

Note that you can change output variable types manually, as described on Page 22. If you choose to do so, the parameter '-O' should not be used with the COPY command.

Doubles Option

When you tell Stat/Transfer to optimize the output variable types, you can also use the 'doubles' option in order to tell Stat/Transfer to put variables with fractional parts into either 'double' or 'float' on output. See Page 20 for details.

The parameter '-OD' tells Stat/Transfer to automatically use doubles where appropriate:

```
COPY  infilename.ex1   outfilename.ex2   -od
```

When this parameter is given, Stat/Transfer will both optimize the variable types and use doubles where needed. The doubles option can only be used when output is optimized.

If you choose to set output variable types manually, the parameter '-OD' should not be used with the COPY command.

Automatic Dropping of Constants from Output File

You can tell Stat/Transfer to automatically drop variables that are constant or missing for a selected subset of data. See Page 21 for details.

In order to tell Stat/Transfer to automatically drop constant or missing variables, the parameter '-OC' is used:

COPY *infilename.ex1*   *outfilename.ex2*   -oc

When this parameter is given, Stat/Transfer will both optimize the variable types and drop variables where needed. Note that the drop-constant option can only be used when the output is optimized.

You can use the doubles parameter and the drop-constants parameter simultaneously, using the parameter '-OCD'.

COPY *infilename.ex1*   *outfilename.ex2*   -ocd

If you choose to set output variable types manually, the parameters '-OC' or '-OCD' should not be used with the COPY command.

Options for Messages

Overwrite Output File Warning

By default, Stat/Transfer will ask permission before overwriting files. If a file exists with the same name as the specified output file, you will be prompted for permission before the file is overwritten.

You can use the parameter '-Y' to suppress the warning messages.

COPY *infilename.ex1*   *outfilename.ex2*   -y

In this case, if an output file already exists with the same name, it will be overwritten with no warning.

Suppress Messages at the Stat/Transfer Prompt

When running a transfer from the Stat/Transfer prompt, by default a number of messages monitoring the progress of the transfer will appear.

The parameter '-Q' will suppress all messages from Stat/Transfer, with the exception of error messages and warnings that a file is about to be overwritten.

COPY *infilename.ex1*   *outfilename.ex2*   -q

Show Messages at the Operating System Prompt

When run from the operating system prompt, Stat/Transfer will only issue error messages and warnings that a file is about to be overwritten.

If you wish to see messages monitoring the progress of the transfer, use the parameter '-V' to get more verbose output.

ST *infilename.ex1*   *outfilename.ex2*   -v

Selecting Cases from the Input File

When you are using the COPY command from the Stat/Transfer prompt or when you are using a command file, you can select cases to be transferred. Transfers that are typed in directly at the operating system prompt will transfer all of the input cases.

You select cases from the input file using the case-selection (WHERE) commands described below. The WHERE statements are entered before the COPY command.

The accuracy of a WHERE statement will be checked only when the COPY command is executed. For this reason, particular care should be taken when typing in variable names.

Case-Selection Expressions

The WHERE statement is used to give the conditions on the variables that will define the subgroup of the data set that you wish to select.

The WHERE statement has the form

WHERE *variable expression* *relational operator* *condition*

where *variable expression* consists of a single variable or an expression involving several variable, *relational operator* is one of the operators listed below and *condition* gives specifications for the variables to be selected.

Relational Operators

All of the usual arithmetic operators [+ - / * ( )] are available for use in this expression, as well as the following relational operators:

=	equals
!=	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
&	and
|	or
,	or (used in a series)
!	not

The modulus operator is also available:

%	the integer remainder after division by the operand following

Internal Variable

An internal variable, '_rownum' is available which allows specific rows or records of the data set to be referenced.

Examples

Examples of selection conditions given by WHERE expressions are:

```
where educ = 12 & rate > .2
```

```
where (income1 + income2)/famsize < 20000
```

```
where income1 >= 20000 | income2 >= 20000
```

```
where acct != 2001
```

```
where name = smith
```

```
where dept = "auto loan"
```

`where  id  %  2 = 0` (which selects all even values of 'ID')

`where _rownum < 200` (which selects rows 1 - 199)

Note that strings used in WHERE expressions need not be enclosed in quotes unless they contain embedded blanks.

Wildcards (* or ?) are available to select subgroups of string variables. For example:

```
where account = ?3*
```

```
where name = mc* | name = mac*
```

Note that when the wildcard '?' is used, it replaces a single character, while the wildcard '*' replaces an unspecified number of characters. Thus the specification '?3*' will select account numbers of any length that have a three in the second place.

The comma operator ',' is used to list different values of the same variable name that will be used as selection criteria. It allows you to bypass potentially lengthy OR expressions when selecting lists of values. For example, the WHERE expression above can be more easily written:

```
where name = mc*,mac*
```

Other examples are:

```
where age = 21,31,41,51,61
```

which will select only the listed ages, and

```
where caseid != 22*,30??,4?00
```

which will select all cases except those id's starting with '22' or four character id's starting with '30' or starting with '4' and ending with '00'.

Missing Values

You can test to see if the value of any variable is missing by comparing it to the special internal variable '_missing.' For example

```
where income != _missing & age != _missing
```

Sampling Functions

Three functions are available for sampling.

The first

samp_rand(*prop*)

allows for simple random sampling. Each case is selected with a probability equal to *prop*.

For example, for a random sample of one tenth of a data set, use:

```
where samp_rand(.1)
```

The second function

samp_fixed(*sample_size,total_observations*)

allows a random sample of fixed size to be drawn. When using this function, the first case is drawn with a probability of *sample_size/total_observations*, and the succeeding *i'th* case is drawn with a probability of (*sample_size - hits) / (total_observations - i)*.

For example, if you had a data set of 1000 cases and wished for a random sample of 25 cases, you would specify:

```
 where samp_fixed(25,1000)
```

Finally, a third function

samp_syst(*interval*)

performs a systematic sample of every *n'th* case after a random start. For instance, to take every 6'th case, use:

```
where samp_syst(6)
```

Expressions are evaluated from left to right. You can thus sample from a subset of your cases by subsetting them first and then sampling. For example, to take a random half of high school graduates, use:

```
where schooling >= 12 & samp_rand(.5)
```

Sampling Seed and Reproducible Samples

The random number generator that provides the basis of these sampling routines is 'rand_port()' in Jerry Dwyer, "Quick and Portable Random Number Generators." *C Users Journal*, June, 1995, pp. 33-44. By default, it is seeded using a permutation of the time of day, and will yield a different sample on each run.

If you need a reproducible sample, you can generate it by using the same seed each time. The seed is entered with the SET command SAMP-SEED and should be a positive integer in the range of one through 2,147,483,646.

Selecting Variables from the Input File

When you are using the COPY command from the Stat/Transfer prompt or when you are using a command file, you can select input variables to be transferred. Transfers that are typed in at the operating system prompt will transfer all of the input variables.

Variables can be selected with one of the commands KEEP, DROP, or TYPES. Only one of these commands can be given and it must be given prior to the COPY command.

KEEP and DROP Commands

The commands KEEP or DROP are used when you wish to select variables but do not wish to change the output variable types assigned by Stat/Transfer.

You can use the SET command DROP-KEEP to reuse the same variable list for more than one transfer. See Page 27.

Specifying a List of Variables.

KEEP or DROP can be followed by a list of variables to be used or omitted, respectively.

KEEP|DROP *variablelist*

Ranges can be given and wildcards are allowed. For example:

```
drop age,sat1-sat10,inc*
```

will drop the variables ‘age’, ‘sat1’ through ‘sat10’ and all fields beginning with ‘inc’.

Using a File to Specify Variables with KEEP or DROP

If you have a file containing a list of variables to be kept or dropped, then instead of a variable list after the KEEP or DROP commands, you can specify the name of the file preceded with an @ symbol. For example:

```
keep @varkeep.lst
```

will read the file ***varkeep.lst*** for a list of variables to be kept. These can be delimited with spaces or new lines.

Note that files used with the KEEP or DROP commands cannot contain the target output types for input variables. Such files must be read with the TYPES command.

The VARS Command

To facilitate the creation of a file containing selected input variables, a command is available that will write a list of variables to a file:

VARS *filename variablelist* [-oc]

where *filenam*e is the name of the input file containing the variables to be transferred (with an extension conforming to the usual conventions) and *variablelist* is the name of the file that is to contain the data list. All of the input variables will be written, one per line, unless the parameter '-OC' has been given, and if variable labels are available, these will also be written as comments in the file.

The parameter '-OC' is used if you wish to drop constant or missing variables from the variable list you are creating. (See Page 21 for details about the 'drop-constants' option.) Note that if you choose this option, and you wish to optimize output variable types, you still need to use the parameter '-O' with the COPY command.

Once the file has been created, you can use your favorite editor to delete the variables that are not to be used with the DROP or KEEP command.

Writing to the Screen

If, for some reason, you wish to have the input variables listed on the screen, use the VARS command without specifying an output file

VARS *filename*

The TYPES Command

If you wish to change the output variable target type manually, you do so using the TYPES command and a file, *vartypelist*, containing the selected input variables and their target output types. (See the next section for a discussion of output variable types.)

The TYPES command has the form:

TYPES *vartypelist*

Note that a @ symbol does not precede the file name.

For example,

```
types vartype.lst
```

will read the file ***vartype.lst*** for a list of variables to be used, along with the target output type chosen. Each variable along with the target output type must be on a separate line.

The VARS Command

The VARS command with appropriate options can be used to write a file giving the output variable types assigned by Stat/Transfer, as well as variable names and labels:

VARS *filename vartypelist* [-t] [-o[d][c]]

where *filenam*e is the name of the input file and *vartypelist* is the name of the file that is to contain the variable list. All of the input variables will be written, unless the parameter 'OC' has been given. They will be written one per line, and if variable labels are available, these will also be written as comments in the file.

The parameter '-T' will write the target output variable type for each input variable to the file, while the '-O', '-OD' and '-OC' parameters will select optimized target types, or optimized types with the 'double' and 'drop-constants' options respectively (see Page 20), before writing the variable list.

Once the file has been created, you can use your favorite editor to delete the variables that are not to be transferred and to change any output types you wish.

Output Variable Types

Systems differ widely in the number and variety of variable types they support. When data are transferred from one file type to another, a variable type in the output format must be assigned to each of the variables being transferred.

Note that with Stat/Transfer, numerical precision is never lost in the transfer process, since all numerical variables are stored internally as double precision floating point numbers and are then written out according to the assigned variable type.

Stat/Transfer will automatically assign output types when you select variables for transfer. You can choose to have the output types optimized and you can change the assigned output types.

Default Types Assigned by Stat/Transfer

When assigning default output variable types, Stat/Transfer attempts to use all of the information at its disposal about the input data variables in order to preserve numeric precision and, at the same time, minimize the size of the output data set.

If you do not choose to have output types optimized, then information about the variables generally comes from the input file "dictionary," which describes the variables. If output types are optimized, then additional information is obtained by examining the values of variables.

When reading numerical variables, Stat/Transfer selects a target output variable type based on the information available to it. This target variable type is not used for internal storage during the transfer, but is simply the preferred output type. If this type is not supported in the chosen output file type, the best approximation will be chosen.

The various target output variable types used by Stat/Transfer are given below.

Stat/Transfer Target Types

byte	one byte signed integer (-128 to 127)
int	two byte signed integer (-32768 to 32767)
long	four byte signed integer
float	four byte IEEE single precision floating point number
double	eight byte IEEE double precision floating point number
date	date stored as serial day number (the number of days since December 30, 1899)
time	fraction of a day (12:00 noon = .5)
date/time	floating point number (integer part - serial day number, fractional part - time)
string	Character string of a maximum length specified by the input file. Internal limit of 255 characters.

If you do not select optimization of target types, then when insufficient information is given about the variables in your input data set to make a specific assignment, Stat/Transfer will generally assign ‘float’ as the output variable type.

Remember that the target type will not necessarily be the actual output type. If the target type assigned to a variable by Stat/Transfer is available as one of the variable types of the output file format, then that type will be used for the output. If the assigned target type is not one of the available output types, then a format of the next larger size will be used.

You can see the target output type assigned to each input variable by using the VARS command with the parameter ‘-T’ as discussed on Page 18. You can then, if you wish, use this file to change the assignments manually.

Optimizing Target Types

Stat/Transfer attempts to produce the smallest possible output data set. By default, information from the data file dictionary will be used to determine the output variable types. Unfortunately, for some input data types, this information is not sufficient to do anything other than set all of the output variable types to ‘float’.

You can choose to have Stat/Transfer make an additional optimization pass through your data to determine more information about each variable.

Stat/Transfer can determine whether any variables can be represented as integers, and, for those cases, it can determine the smallest integral type that can be used to represent the data. Further, if a variable cannot be represented by an integral type, Stat/Transfer can automatically determine whether it can be represented by a float instead of a double without a loss of information. Information on the maximum length of string variables is also accumulated, so that these can be stored in variables of the smallest possible length.

In order to tell Stat/Transfer to automatically optimize the output variable types, the parameter ‘-O’ is used, either with the COPY command or with the VARS command when the ‘-T’ parameter is used.:

COPY *infilename.ex1* *outfilename.ex2* -o

VARS *filename vartypelist* -t -o

When this parameter is given, Stat/Transfer will make an optimization pass when the output file type will benefit from optimization.

If you have given the parameter ‘-O’ with the VARS command and then use the resulting file with TYPES, you should not give the parameter with the COPY command.

Doubles Option

When you tell Stat/Transfer to optimize the output variable types, you can also use the ‘doubles’ option in order to tell Stat/Transfer to put variables with fractional parts into either ‘double’ or ‘float’ on output.

If you choose to use doubles, Stat/Transfer will still evaluate each variable to see if it can be represented as a ‘float’ without a loss of information and will put only those variables that require it into a ‘double’. However, unless your data are measured with more than eight or nine digits of precision (survey data, for example, never are), this is an idle exercise and you should not use the ‘doubles’ parameter.

In order to tell Stat/Transfer to automatically use doubles where appropriate, the parameter ‘-OD’ is used:

COPY *infilename.ex1* *outfilename.ex2* -od

VARS *filename vartypelist* -t -od

When this parameter is given, Stat/Transfer will both optimize the variable types and use doubles where needed. Note that the doubles option can only be used when the output is optimized.

If you have given the parameter ‘-OD’ with the VARS command and then use the resulting file with TYPES, you should not give the parameter with the COPY command.

Automatic Dropping of Constants from Output File

You can tell Stat/Transfer to automatically drop variables that are constant or missing for a selected subset of data.

In order to tell Stat/Transfer to automatically drop constant or missing variables, the parameter ‘-OC’ is used:

COPY *infilename.ex1* *outfilename.ex2* -oc

VARS *filename vartypelist* -t -od

When this parameter is given, Stat/Transfer will both optimize the variable types and drop variables where needed. Note that the drop-constant option can only be used when the output is optimized.

This feature is useful when the part of a data set selected for transfer contains variables such as a pregnancy variable when only male subjects are selected or variables in yearly surveys where the same questions do not appear for each year.

This feature is not likely to be used often, but is extremely valuable when it is needed, since if the data set has a large number of variables, it can be exceedingly tedious to select only the meaningful ones manually.

If you have given the parameters ‘-T’ and ‘-OC’ with the VARS command and then use the resulting file with TYPES, you should not use any of the optimization parameters with the COPY command. However, if you use ‘-OC’ while creating a file to use with the KEEP or DROP commands and then wish to automatically optimize the output variables, you do need to give the appropriate optimization parameter after the COPY command.

The Parameter -OCD

You can use the `doubles’ parameter and the `drop-constants’ parameter simultaneously, using the parameter ‘-OCD’.

COPY *infilename.ex1* *outfilename.ex2* -ocd

VARS *filename vartypelist* -t -ocd

If you have given the parameter '-OCD' with the VARS command and then use the resulting file with TYPES, you should not give the parameter with the COPY command.

Manually Changing Output Variable Types

Stat/Transfer will assign target types for output variables as discussed on Page 19, either with or without optimizing. You can, if you wish, change the assigned output types for selected variables.

In most cases it is appropriate to accept the output type that Stat/Transfer chooses. However, there may be times when you wish to specify the output types for some variables, since sometimes Stat/Transfer does not have enough information to make the best assignment of an output variable.

For example, if social security numbers are stored as numbers instead of strings in the input file, Stat/Transfer will generally convert them into floats on output, possibly resulting in the loss of several digits in the output data set. You can avoid this loss of key values by specifying that social security numbers be stored as longs or doubles on output.

In some transfers, you may prefer a larger data set than Stat/Transfer chooses, with more precision in some or all of the variables. For instance, in the absence of specific information to the contrary, Stat/Transfer will usually chose 'float' (four-byte, floating-point) format for numeric variables. However, you may wish to convert these into double precision numbers if you know that they represent large monetary amounts.

In other cases, you may be able to create a smaller data set than Stat/Transfer chooses. For example, if you know that your data represent small integers, you may wish to put them into 'byte' or 'integer' variables.

Handling Mixed Data

If you have mixed data in which some variables need doubles and others do not (for example, you might have precisely measured dollar amounts, which should be in doubles, along with scales of survey items, which should be in floats) you should use the 'optimize' parameter '-O' in order to designate integers for the right variables and then manually designate floats and doubles to reflect the appropriate level of measurement for each variable.

Determining the Assigned Target Types for Output Variables

In order to determine the target output type assigned by Stat/Transfer, first use the VARS command with the parameter '-T' to create a file which contains the names of the variables in the input file and also their assigned output type, as discussed on Page 18.

For example, if you wish to transfer data from the file ***indata.dta***, then the command

```
VARS indata.dta  variabletype.lst  -t
```

will create a file ***variabletype.lst*** with the variables from ***indata.dta*** listed, along with their target output types.

If you wish to have Stat/Transfer automatically optimize the target output types before creating the file, or if you wish to use the 'doubles' or 'drop-constant' options before creating the file, then use the parameters '-O', '-OD', '-OC', or '-OCD' with the VARS command and the parameter '-T'.

VARS *filename vartypelist* [-t] [-o[d][c]]

If you use these parameter with the VARS command, do not use them with the COPY command.

Selecting Variables to Be Transferred

Use any editor to delete the lines in the *vartypelist* file corresponding to the variables that you do not wish to transfer.

The resulting file should still have one line for each variable that you do wish to transfer.

Changing the Assigned Output Types

Again, using any editor, you can change the output variable type for any variable listed in the *vartypelist* file.

Valid types are string, byte, int, long, float, double, date, time, and date/time, as shown in the table on Page 19. You should be careful not to choose a smaller type than that chosen by Stat/Transfer unless you are sure you know more about your data than Stat/Transfer does.

Output variable types can be changed freely for ASCII files and worksheet files.

For all other file formats, you can change freely among the numeric types of 'byte', 'integer', 'long', 'float' and 'double' and you can change among the time types. However, do not make changes such as string to number, or number or string to date.

Remember that you are selecting a "target" type. If the output data format does not support the specific type you have selected, then Stat/Transfer will use the best match to the type you have selected.

You can determine the output variable types supported by a given output file type by consulting the table given in the section describing that file format.

The TYPES command

Once you have specified target output variable types in the *vartypelist* file, you then use the TYPES command to tell Stat/Transfer to use that file to read input variables and target output variable types.

TYPES *vartypelist*

For example,

```
types variabletype.lst
```

will read the file ***variabletype.lst*** for a list of variables to be kept, along with the target output types chosen.

The only checking that is done at the time the TYPES command is given is for the existence of the file, so that editing needs to be done with care.

This command should be given before the COPY command. The file with the variable types will be processed during the execution of the COPY command.

Options Set with the SET command

The general format of the command used to set an option is:

SET *optionname value*

All SET commands should be given before the COPY command.

A list of available options can be obtained on the screen by typing either of the following:

```
help set
help set all
```

Note that the SET options are not available when a transfer is run by typing it in directly at the operating system prompt, unless you use the startup options file discussed below.

Setting Options

If you wish, you can type each SET command in at the Stat/Transfer prompt before you give the COPY command. However, if several options are to be set, this can be tedious and error-prone. Instead, a list of all of the available options and their default settings can be written out to a file, *filename*, by typing:

```
SET LIST filename
```

Using the resulting file, settings can be conveniently edited and then used in a command file (see Page 39) or a startup options file.

Startup Options File

You can set options globally and automatically at startup by putting them in a file called ***profile.stc***, which should either be in the current directory or in same directory as your Stat/Transfer program file.

If the program ***profile.stc*** is found, it will be executed as Stat/Transfer starts up.

Available Options

The following options are available for use with the SET command. Case does not matter when they are entered. They are discussed in detail in the sections that follow.

General Options

WRITE-OLD-VER	(Y/N)	Write older versions of output file
PRESERVE-CASE	(Y/N)	Preserve name and label case if possible
SAMP-SEED	(*Value*)	Seed for sampling functions
USER-MISS	(All/First/None)	Honor user-missing values
CENTURY-CUTOFF	(*Value*)	Century cutoff for two digit years

DROP-KEEP	(Clear/Save)	Reset variable selection statement
NUMERIC-NAMES	(Y/N)	Write new, numeric variable names
BYTE-ORDER	(Native/HL/LH)	Set byte order for output SPSS and S-PLUS files

Date and Time Options for ASCII Files

DATE-FMT-READ	(*Format string)*	Format for reading dates from ASCII files
TIME-FMT-READ	(*Format string)*	Format for reading times from ASCII files
DATETIME-FMT-READ	(*Format string)*	Format for reading date/time from ASCII files
DATE-SCAN-FMT	(*Format string)*	Format for scanning ASCII files for dates and times
DATE-FMT-WRITE	(*Format string)*	Format for writing dates in ASCII files
TIME-FMT-WRITE	(*Format string*)	Format for writing times in ASCII files
DATETIME-FMT-WRITE	(*Format string)*	Format for writing date/time in ASCII files

ASCII Options-Read

DELIMITER-RD	(Autosense/ Comma/Tab/ Space,/Semicolon)	Delimiter for reading ASCII files
ASCII-RD-VNAMES	(Autosense/First-row/ Make-up)	Field names when reading ASCII files
NUM-MISS-RD	(*Missing value characters/* None)	Numeric missing value when reading ASCII files
QUOTE-CHAR-RD	(*Character /*None*)*	Quote character used when reading ASCII files
MAX-FIELDS	(*Number*)	Maximum fields to be read when reading ASCII files
MAX-WIDTH	(*Number)*	Maximum input line width when reading ASCII files
MAX-LINES	(*Number*/All)	Maximum ASCII lines to examine for variable type
DECIMAL-POINT	(*Character*)	Decimal point for reading ASCII files
THOUSANDS-SEP	(*Character*)	Thousands separator for reading ASCII files

ASCII Options-Write

DELIMITER-WR	(Comma/Tab/ Space/Semicolon)	Delimiter for writing ASCII files
QUOTE-CHAR-WR	(*Character*/None)	Quote character used when writing ASCII files
NUM-MISS-WR	(*Missing value characters/* None)	Numeric missing value when writing ASCII files
WRITE-FIELD-NAMES	(Y/N)	Write field names to ASCII

SAS Options

READ-SAS-FMTS	(Y/N)	Read value labels from a SAS data file
READ-FMT-NAME	(*Filespec*)	Name of SAS value label data file
WRITE-SAS-FMTS	(Y/N)	Write Proc Format program for SAS
WRITE-FMT-NAME	(*Filespec)*	File name for Proc Format program

Worksheet Options

WKS-NULL-ROWS	(Stop/Skip/Use)	Treatment of worksheet null rows

WKS-NAME-ROW	(Autosense/First-row/ No-name/*Number*)	Reading of worksheet variable names
WKS-DATA-RANGE	(Autosense/*Name/ Coordinates*)	Specifying a data range as input

General Options

Write Older Version of Output File

WRITE-OLD-VER (Y/**N**)

By default, a particular output file type will be written using the latest version of that file format supported by Stat/Transfer. For example, Stata Version 7 files will be written if Stata is specified as the output file type.

To write a file for the previous version of a particular file type, use the SET command

```
set write-old-ver y
```

For Stata output, for example, this option would tell Stat/Transfer to write a Stata Version 6 output file.

Preserve Variable Name and Label Case if Possible

PRESERVE-CASE (Y/**N**)

Stat/Transfer always follows the variable-naming rules of the output file type and will convert input names so that they will conform to those rules. It also, by default, tries to convert labels in keeping with the “spirit” of the target package. This means that for packages such as S-Plus and Stata, Stat/Transfer will write out variable names and labels in lower case.

For S-Plus and Stata only, if you want to override this behavior, use the SET command

```
set preserve-case  y
```

and the case of your input variables will be preserved on output.

Seed for Sampling Functions

SAMP-SEED (*Value*)

By default, the sampling functions in WHERE expressions will generate a starting seed randomly based on the clock time. This means that each time you run a transfer on a given file you will select a different sample. If, in contrast, you need a reproducible sample, you can enter a seed for the random sampling process. The seed should be a positive integer in the range of one through 2,147,483,646.

Honor User Missing Values

USER-MISS (**All**/First/None)

You have control over the way missing values are treated for input files containing more than one type. At present the USER-MISS options apply to SPSS files (both Data and Portable) and OSIRIS. The options are ‘All’, ‘First’ or ‘None’.

Some statistical systems distinguish between “system missing,” such as the result of a divide by zero, and “user-missing,” a numeric value which is defined as a missing value by the user. Further, particularly in survey research, distinctions are made between user-defined missing values that represent structurally missing data (such as answers to pregnancy history questions from male respondents), and those that represent categories of non-response or simply the failure of the interviewer to properly collect the data.

Conventionally, zero is used to represent "inapplicable" missing values, and higher numbers are used to represent such responses as "don't know," "refused" and "not ascertained". While inapplicable data is analytically equivalent to "system missing", there can be legitimate research interest in the patterns of non-response represented by the other categories of missing data.

All

In prior versions of Stat/Transfer, when multiple missing values were allowed (as in SPSS, for example) they were always mapped into a single missing value on output. This is still the default behavior, corresponding to specifying 'All'.

First

If you specify 'First', the first user-defined missing value will be mapped to the system missing value and the rest will be transferred intact to the target data set. This will often be the most useful of the options, since it will allow tabulations in the target package of patterns of non-response.

None

If you specify 'None', then all of the user-defined missing values will be transferred retaining the input value in the target data set.

Century Changeover for Two Digit Years

CENTURY-CUTOFF (*Value*)

The default value is '20'.

If you are reading two digit years and some or all of them fall outside of the twentieth century, you can use this option to control how they are read. Any two-digit year less than the changeover year will have the first two digits of the complete four digit year set to 20. Any year greater than or equal to the changeover year will have the first two digits set to 19.

The default for the option is '20', so that the changeover year from one century to another is 1920. Thus the date 1/1/01 will be interpreted as January 1, 2001, while the date 1/1/31 will be interpreted as January 1, 1931.

If your data refer to dates earlier than 1920, such as birth dates, you will need to override the default behavior and specify a different changeover year. If, for example, you specify '00', all two digit dates will be interpreted as years in the twentieth century.

Reset Variable Selection Statement

DROP-KEEP (Clear/Save)

If data are transferred from more than one input file during a single session, then you need to specify the variables that are to be transferred from each file, using the KEEP or DROP variable selection command. The DROP-KEEP options of the SET command allow you to reuse or clear the variable selection command.

Input Files Specified Separately

If the input files are specified with separate COPY commands (that is, without using wildcards), then the default behavior is that you must give a separate KEEP or DROP variable selection statement before each COPY command. This corresponds to the option 'Clear' for DROP-KEEP.

However, if the same variables are to be transferred from each file, you can specify that the same KEEP or DROP command apply to all input files that follow, until another KEEP or DROP command is encountered. To do so use the SET command

```
set drop-keep save
```

Input Files Specified with Wildcards

If the input files are specified with wildcards, then the default behavior is that the KEEP or DROP variable selection command you give before a COPY command applies to all of the files specified in that command. Thus the default when files are specified with wildcards is 'Save'.

If you set the DROP-KEEP option to 'Clear', then the variable selection statement will apply only to the first file. All of the other files given by the wildcard specification will transfer all of their variables.

Note that the default changes depending on whether or not the input file specification contains wildcards. The default is "Clear' for no wildcards and 'Save' for wildcards.

Write New, Numeric Variable Names

NUMERIC-NAMES (Y/**N**)

When you go from one format to another, by default Stat/Transfer will create legal variable names for you, based as much as possible on the original names. In particular, when you transfer from a system such as JMP, which allows long variable names with embedded spaces, to a system such as SAS, which restricts variable names to eight characters, by default Stat/Transfer will truncate for you. These truncated names often have little resemblance to the names you started with. Therefore, Stat/Transfer, when possible, uses the variable names as variable labels, so that the original names are available.

You can override the default behavior. If you use the SET command

```
set  numeric-names  y
```

then, instead of the default variable names, Stat/Transfer will create new variable names of the form V1...VN, while still using the original variable names as labels. This is chiefly useful when dealing with truncated names. If your output system supports variable labels, it is sometimes better to use this option and have Stat/Transfer simply create numeric names. You can then use the variable labels for the description.

Set Byte Order for S-PLUS and SPSS Output Files

BYTE-ORDER (**Native**/HL/LH)

By default, Stat/Transfer will write files in the byte order of the machine on which it is running. DEC Alpha and Intel processors are low-high byte order, other Unix machines are high-low. You can change this order to write a file appropriate for another machine.

For example, if you are running on a SUN machine and wish to produce a file suitable for an Alpha machine, you would use the SET command

```
      set byte-order lh
```

To produce files for Windows applications, choose a low to high byte order.

Date and Time Options for ASCII Files

Date/Time Formats - Reading

DATE-FMT-READ (*Format string)*

TIME-FMT-READ (*Format string*)

DATETIME-FMT-READ (*Format string*)

DATE-SCAN-FMT (*Format string*)

You can control how Stat/Transfer reads dates and time. In most cases, users will not need to change the default settings in order to read date and time variables. However, if you do need to do so, you provide a general "scanning" input format string with the DATE-SCAN-FMT option of the SET command. This modified format is used when an ASCII file is opened for reading. The input format given in

```
      set date-scan-fmt  formatstring
```

is used in the initial look at the file, which will determine variable types and also specific formats for dates, times, and date/time variables. The default input format string is constructed so that it will decipher a number of different date and time possibilities.

The separate input format strings for date, time and date/time variables are given with the DATE-FMT-READ, TIME-FMT-READ, and DATETIME-FMT-READ options of the SET command and must match those given in the general scanning input format. (These formats are used when the data file is being used in a transfer and allow for much more efficient reading of the file.)

The input format strings given with the DATE-FMT-READ, TIME-FMT-READ, and DATETIME-FMT-READ options of the SET command are used to convert character strings to date/time variables. The format strings are read from left to right. If a width is given explicitly for a particular variable, that will be used when reading the character string. Otherwise, the width will be determined by the presence of delimiter characters. Characters in the list below allow characters in a string to be skipped, if necessary.

If the entire input string is not matched by the format string or if the resulting time or date is not valid, the variable will be set to missing.

Each date and time part of the input format, as well as some special characters, have the form '%*char'* or '%*Xchar*', where the modifier *X* is used to determine field widths.

The two different cases of '%*Xchar*' are:

%*numberchar* When a number precedes the specification character, *char*, it specifies the field width to be used, so that the next *number* characters in the input string are scanned for the specified date or time.

%*:delimchar* If there is a colon and any single character, *delim*, preceding the specification character, the field to be read is taken to be all the characters up to but not including the given delimiter character, *delim*. The delimiter itself is not scanned or skipped by the format, and therefore must be entered explicitly in the input format or explicitly skipped. (Note that the modifier *:delim* need not be used routinely, since numeric and alpha formats will automatically stop reading when they reach a delimiter.)

The characters used to create the input format strings are listed below. Anything to be read from the input character string that is not in the list below, such as commas or other delimiters, must be given explicitly in the input format. White space (spaces, tabs, carriage returns, and so on) is ignored in the input format string.

%c	skip a single character (see also %w)
%Nc	skip N characters
%$c	skip the rest of the input string
%d	input day of the month
%H	input hour
%m	input month, as integer or as alpha string. (If alpha string, case does not matter, and any substring of a month that distinguishes it from the other months will be accepted.
%M	input minute
%n	input milliseconds
%N	input milliseconds or tenths or hundredths of seconds. (If no field width is given and the input string has a field width of three, then input will be milliseconds. A field width of 1 (either given explicitly or inferred from the input string) will cause input of 10ths of a second; a width of 2 will cause input of 100ths of a second.)
%p	input strings defining 'am' and 'pm' (Matching is the same as for months.)
%S	input seconds
%w	skip a white space delimited word (see also %c)
%y	input year. (If less than 100, the century changeover year is used to determine the actual year.)
%Y	input year as found in the input string.
%%,%[,%]	input the '%', '[', and ']' characters from the input string.
[...]	optional specification. (Text and specifications within the brackets will be read if present in the input string, but need not be there.)

The default format string for scanning times in text files is:

```
[%m[/]%d[/][,]%y] [%H:%M[:%S[.%N]][%p][[(]%3c[)]]]
```

which will recognize such diverse strings as:

"May 18 1945", "May 18, 1945", "5/18/45", and "05/18/45 2:16 PM"

Date and Time Formats - Writing

DATE-FMT-WRITE (*Format string*)

TIME-FMT-WRITE (*Format string*)

DATETIME-FMT-WRITE (*Format string*)

Stat/Transfer gives you considerable control over how dates and times written to output ASCII files (see above for controlling how dates and times are read.) You can control the formatting for date values, time values and combined date/time values using the DATE-FMT-WRITE, TIME-FMT-WRITE, and DATETIME-FMT-WRITE options of the SET command.

Output formats strings that you supply are used to convert date and time values to character strings. Each date or time part of the output format string has the form '%char'. A leading zero causes the value to printed with leading zeros. For example '%0d' will print the day of the month with a leading zero.

The characters below are used to create the output format strings. Anything to be printed in the output character string that is not in the list below, such as commas, spaces or other delimiters, must be given explicitly in the output format string.

%a	abbreviated weekday
%A	full name of weekday
%b	abbreviated name of month
%B	full name of month
%d	day of the month (1 - 31)
%D	day of the year (1 - 366)
%H	hour (24 hour clock) (0 - 23)
%I	hour (12 hour clock) (1 - 12)
%m	month as number (1 - 12)
%M	minutes (0 - 59)
%N	milliseconds (0 - 999)
%1N	tenths of seconds (0 - 10)
%2N	hundredths of seconds (0 - 99)
%p	'am' or 'pm'
%S	seconds (0 - 59)
%y	year as two digits
%Y	year as four digits
%%	% character

The default formats for converting dates and times to strings are:

Date:	`%m/%d/%Y`	(5/18/1945)
Time:	`%0H:%0M:%0S`	(14:05:48)
Date/Time:	`%m/%d/%Y %0H:%0M:%0S`	(10/1/1990 02:20:09)

ASCII Options - Reading

Delimiter for Reading ASCII Files

DELIMITER-RD (**Autosense/**Comma/Tab/Space/Semicolon)

By default, Stat/Transfer will automatically sense which delimiter to use when reading ASCII files. You can set the delimiter explicitly with the DELIMITER-RD options of the SET command. For example, if the delimiter is a comma, you would use the SET command

```
set delimiter-rd comma
```

Field Names when Reading ASCII Files

ASCII-RD-VNAMES (**Autosense/**First-row/Make-up)

By default Stat/Transfer will sense whether the first line of your input data set contains field names or data. You may, if you wish, explicitly override this default and use the ASCII-RD-VNAMES options to tell Stat/Transfer to expect either data or field names as the first line of the data set.

The SET command

```
set ascii-rd-vnames first-row
```

specifies that the first row contains field names.

The option 'make-up' specifies that the first row contains data and that Stat/Transfer should assign field names 'col1', 'col2', ... 'col*n*' to the data.

Numeric Missing Value when Reading ASCII Files

NUM-MISS-RD (*Missing value characters/***None***)*

By default, Stat/Transfer will interpret two consecutive delimiters as a missing value. Using the NUM-MISS-RD option, you can specify a string that will be interpreted as a missing value when Stat/Transfer reads ASCII files. For example, your input data set may use the string 'NA' to represent missing values or it may use a period.

If you give the SET command

```
set num-miss-rd  na
```

then 'na' in the input data set will be interpreted as a missing value.

String Quote Character when Reading ASCII Files

QUOTE-CHAR-RD (*Character* /None*)*

This option allows you to set the character that is used to enclose string fields in the input data set. The default character is set as double quotes.

You can use this command to choose the appropriate character for the input data. For example, if the quote character is a single quote, the SET command would be

```
set quote-char-rd '
```

However, if string variables are not enclosed by any character, you can leave this option set at the default double quote.

Maximum Number of Fields to Be Read

MAX-FIELDS (*Number*)

By default in Stat/Transfer, the maximum number of fields in a delimited data set is set at 256. If you have more fields, set the value of this option to a higher value.

Maximum Input Line Width when Reading ASCII file

MAX-WIDTH (*Number)*

The maximum width of an input line defaults to 4096 characters in Stat/Transfer. If your lines are longer you should set this value to a higher number.

Maximum Number of Lines to Examine for Variable Type

MAX-LINES (*Number/***All**)

Stat/Transfer first reads your ASCII data to determine what type of variable is present in each delimited position. By default it will read your entire data set. If you data are consistent, so that the first few lines suffice to show each variable type, and your data have enough rows that it actually takes more than a few seconds to examine them all, you might want to set this option to a numeric limit, such as 50.

Decimal Point for Reading ASCII Files

DECIMAL-POINT (*Character*)

The period is the default character used for the decimal point. If your data set uses a symbol other than the default period to indicate the decimal point in a number (a comma, for example), specify the character with the DECIMAL-POINT option.

Thousands Separator for Reading ASCII Files

THOUSANDS-SEP (*Character*)

The comma is the default thousands separator. If your data set uses a symbol other than the default comma to mark thousands in a number (a period, for example), specify the character with the DECIMAL-POINT option.

ASCII Options - Writing

Delimiter for Writing ASCII Files

DELIMITER-WR (**Comma**/Tab/Space/Semicolon)

By default, ASCII files will be written with a comma. You can change this behavior with the DELIMITER-WR option.

For example, to write files with tabs as the delimiter, use the SET command

```
set delimiter-wr tab
```

String Quote Character when Writing ASCII Files

QUOTE-CHAR-WR (*Character*/None)

The character specified here will be written before and after string variables on output. The default value is a double quote.

This character is only strictly necessary if your fields have an embedded delimiter. If you enter 'None' here, string fields will not be enclosed by any character.

Numeric Missing Value when Writing ASCII files

NUM-MISS-WR (*Missing value characters*/**None**)

This option is used to specify the string that will be used to represent missing values when Stat/Transfer writes ASCII files. By default, nothing will be used for missing values.

For example, you may want to create an output file for a program that expects a specific missing value, such as a period for SAS. The SET command would then be

```
set num-miss-wr .
```

Write Variable Names in First Row when Writing ASCII Files

WRITE-FIELD-NAMES (**Y**/N)

By default, variable names will be written in the first row of an output ASCII data file. If you do not wish to have variable names written, then use the SET command

```
set write-field-names n
```

SAS Options

Read Value Labels from a SAS datafile

READ-SAS-FMTS (Y/**N**)

READ-FMT-NAME (*Filespec*)

As described on Page 75, Stat/Transfer can read value labels from a SAS data set produced by PROC FORMAT using the 'cntlout' keyword.

By default, Stat/Transfer will not look for this file. If you have generated a file that contains value labels and wish to use it, you should use the SET command

```
set read-sas-fmts y
```

Stat/Transfer will look in the same directory as the input data set for a file named ***sas_fmts.****ext,* where *.ext* is the same as the input file extension. If you would like to use a file located somewhere else or one with a different name, you must use the SET command

```
set read-fmt-name filespec
```

You can type in a complete file specification, or you can use either of the macros below as part of the file specification.

%ipath%	The path, including the directory, of the input file
%*iname*%	The name, without the extension, of the input file.
%*iext*%	The extension, without the dot, of the input file

Write a PROC FORMAT Program for SAS

WRITE-SAS-FMTS (Y/**N**)

WRITE-FMT-NAME (*Filespec*)

Stat/Transfer can optionally write a PROC FORMAT program that you can use to import value labels from other systems into SAS. By default the program is not written. To turn on this option use the SET command

```
set write-sas-fmts y
```

The program file will by default be written to the same directory as your output data file, with the same name as your output file, but with the extension ***.sas***. You can change this default with the set command

```
set write-fmt-name filespec
```

You can type in a complete file specification, or you can use either of the macros below as part of the file specification.

%*opath*%	The path, including the directory, of the output file
%*oname*%	The name, without the extension, of the output file
%oext%	The extension, without the dot, of the output file

Worksheet Options

Reading of Worksheet Null Rows

WKS-NULL-ROWS (**Stop**/Skip/Use)

This SET command applies to cases where an entire worksheet page is to be read. It will not affect cases where specified ranges are used as input.

The default behavior is 'Stop'. Stat/Transfer begins reading a worksheet page at the first non-blank cell. With default behavior, it will then read data until it encounters an entirely blank line and will end the transfer at this point.

If the option 'skip' is used, Stat/Transfer will read the entire worksheet page, searching for further non-blank rows, but will not write out null lines that it finds.

If the option ‘use’ is selected, then all rows of the input worksheet page, including blank rows, will be written to the output file. This option may return unexpected blank rows from the end of a worksheet page that contains formatting but no data.

Variable Name Row

WKS-NAME-ROW (**Autosense**/First-row/No-name/*Number*)

By default, Stat/Transfer will attempt to autosense whether or not the data in the first non-blank row (or the first row of a specified range) are variable names or the first row of data. It does so by looking for at least one column in which there is a string in the first row and a number in the second. If this behavior is inappropriate for your worksheet (for example, if you have only string data), you can override it.

You can specify one of the following options using WKS-NAME-ROW:

First-row

This option will take the values in the first non-blank row as the field names even if there is no change to a numeric type in the second row.

No-name

This option will treat the first non-blank row as data. Stat/Transfer will assign variable names ‘col1’ through ‘col*n*‘.

Number

This option will take the field names from a specified row. If, for example the field names are in the third row, the SET command would be

```
set wks-name-row 3
```

Specifying a Data Range

WKS-DATA-RANGE (**Autosense**/*Name/Coordinates*)

You can choose different ranges to be read in input worksheets by using this option. When a range has been specified, Stat/Transfer will read of the all rows of the specified range and write them to the output data file.

Autosense

The default behavior is ‘Autosense’. Stat/Transfer will read to the first non-blank cell and use that as the upper left quadrant of the data range. It will then read data until it encounters an entirely blank line. You can change the default behavior by specifying a range, either by name or by giving explicit coordinates.

Name

If you wish to use a named range, then give the name after the WKS-DATA-RANGE option. For example, for a range named ‘QRT1’, the SET command would be

```
set wks-data-range qtr1
```

Coordinates

If you wish to, you can specify a worksheet range by giving the coordinates that define the range (such as C3:F280). The SET command would then be

```
set wks-data-range c3:f280
```

Other Available Commands

Commands are available at the Stat/Transfer prompt which are the same as operating system commands. You can also get help from the prompt and, if you wish, you can produce a log of your transfer session.

Running Programs and Shell Commands

You can run other programs or give shell commands from the Stat/Transfer prompt. In order to do so, type ‘shell’ or ‘!’ followed by the command.

For example, to delete a file, type either of the following:

```
shell rm sasdata.xtp
```

or

```
! rm sasdata.xtp
```

You can give a series of commands. For example, you might use the following set of commands, allowing you to zip or unzip files with commands given to the Stat/Transfer command processor. .

```
// unzip a SAS file
!unzip  sasdata.zip  mysas1.sd2
// transfer it to an SPSS file
copy  mysas1.sd2  myspss1.sav
// zip your new SPSS file
!zip  myspss1.zip  myspss1.sav
// delete the SPSS file
!rm myspss1.sav
// delete the SAS file
!rm mysas1.sd2
```

where ‘//’ indicates to Stat/Transfer that what follows is a comment and ‘!’ has been used instead of ‘shell’.

A sequence of commands such as these can be entered directly at the Stat/Transfer prompt or stored in a file for batch execution.

Changing Directory and Listing Files

The commands for changing directories, ‘cd’, and for listing files in a directory, ‘ls’, are used so often that these are exceptions to the above and do not need to be preceded by either ‘shell’ or ‘!’. For example,

```
ls *.xls
```

will list all files with the extension ***.xls*** in the current directory.

Stat/Transfer Help

You can issue the following commands to get help on various topics:

```
help commands
help copy
help formats
help running
help set
help set all
help set general
help set date
help set sas
help set ascii
help set worksheet
```

To get a list of the available help commands, type:

```
help
```

Logging Stat/Transfer Sessions

You can log your transfer session to a file by typing:

LOG USING *filename*

where *filename* is the name of the log file.

All of the commands that you enter and all messages from Stat/Transfer will be written to the file.

Quit

You can terminate your Stat/Transfer session by typing:

```
quit
```

or

```
q
```

Command Files

You can enter Stat/Transfer commands in two ways:

- Interactively at the Stat/Transfer prompt or at the operating system prompt.
- In a command file, to be executed either from the Stat/Transfer prompt or from the operating system prompt.

When you wish to run batch jobs, you must use command files.

Constructing Command Files

You can create Stat/Transfer command files as ASCII documents. Type commands in the document just as you would type them at the Stat/Transfer prompt.

Command File Extensions

Stat/Transfer assumes that command files have the extension *.**stc*** (**S**tat/**T**ransfer **C**ommand). You can use files with different extensions only if you execute them from the Stat/Transfer prompt.

If you are executing a command file from the Stat/Transfer prompt, you need not give an extension unless the file does not have the extension *.**stc***.

If you wish to run a command file from the operating system prompt, it must have the extension *.**stc*** and the extension must be given.

Executing Command Files

Executing Command Files from the Stat/Transfer Prompt

You can execute a command file at the Stat/Transfer prompt by typing:

EX *commandfilename*

If you enter *commandfilename* without an extension, an extension of *.**stc*** is assumed by Stat/Transfer.

A command file named ***weekly.stc*** could be executed by typing

```
ex weekly
```

Executing Command Files from the Operating System Prompt

You can execute a command file at the operating system prompt by typing st   *commandfilename.stc*

For example, a command file, ***repeat.stc***, will be executed if, at the operating system prompt, you type:

```
      st repeat.stc
```

Note that the extension, which is explicitly given, must be ***.stc***. If the file is not in the Stat/Transfer program directory, the complete path to it must be given. Also make sure that the file name is in the correct case.

Variable Naming and Limits

This section contains general information about Stat/Transfer's handling of variables and about some general program limitations. Specific information for each of the various file formats is given in the next section.

Variable Names

Stat/Transfer will, by default, ensure that variable names are legal in the target data set and are unique. It does this by adding special characters when needed, changing case where appropriate, substituting underscores (or another legal character) for invalid characters and by truncating variable names to an acceptable length. It then eliminates duplicates created by this process by appending a number to the end of the second and higher instances of duplicate variable names.

The original variable names will, if possible, be retained as variable labels.

If you choose the option 'Y' for the SET NUMERIC-NAMES command, then instead of the default variable names, Stat/Transfer will create new variable names of the form V1...VN. This is chiefly useful when dealing with truncated names, which often have little resemblance to the names you started with. If your output system supports variable labels, you may wish to use this option to assign numeric names for your variables. You can then use the variable labels for description.

Limitations on the Number of Variables

Stat/Transfer will check whether the number of variables that have been selected exceeds the maximum permitted by the target program and will inform you if too many variables have been selected.

Internal Limitations

The maximum width of an alphanumeric variable is 255 characters.

Variable names are limited internally to a maximum length of 32 characters.

Data are generally processed one case at a time, so the number of cases is not limited. Exceptions to this are the worksheets and file formats that must be transposed, such as S-Plus. These transfers are limited by available virtual memory. It is unlikely, however, to be a problem.

Return Transfers to the Original Format

Because systems vary widely in the types of variables they support and the information they store about them, you must be careful when transferring data from one format to another and then transferring back to the original format. Stat/Transfer will always transfer numeric values without losing accuracy. However, other information about variables may be lost.

For example, SPSS maintains value labels and up to three types of missing value for each variable. If you transfer from an SPSS data set to dBASE, the value labels will be dropped and missing values will be translated into a single dBASE internal format. If you were then to transfer your data back to SPSS, missing numbers would all be translated into the SPSS system missing value, not the original values, and because dBASE does not store value labels in the data set, the value labels would be lost.

Because many systems do not explicitly support dates, you should be particularly cautious when planning two-way transfers of date variables. Stat/Transfer will generally convert date variables into MM/DD/YY strings when transferring to systems that do not explicitly support dates. This makes the date intelligible in the destination system. However, when you transfer back to the source system, Stat/Transfer has no way of knowing that these strings are dates and will treat them simply as character variables.

If you are planning such a two-way transfer, you may wish to treat date data as numbers rather than as dates. Stat/Transfer stores dates internally as the number of days since December 30, 1899. Thus, if you wish, you can choose to put dates directly into your destination data set as numbers in this form by selecting the proper target type (see Page 22.) If you do so, these will survive a two-way transfer involving a system that does not explicitly support dates.

Technical Support

We provide support only to registered users. You must return your registration card, or register online through our web site, if you wish to receive support. Our web site address is **http://www.stattransfer.com.**

Before you call for support, please look in this manual and see if the solution to your problem can be answered by this aid. Be sure to check the “Frequently Asked Questions” section on Page 89 of the manual.

You can also check to see if your problem is addressed in the **Support** section of our web site.

If you are unable to resolve your problem with the help tools available to you, please contact us by e-mail at **support@circlesys.com** or call us at **(206) 682-3783**. In addition, you can reach us by fax at **(206) 328-4788**.

When you contact us, please let us know the nature of your problem and any error messages you encountered. It is also helpful if you write down the exact version of Stat/Transfer that you are using. This will be shown on the screen at when you start Stat/Transfer.

Our general e-mail address, should you want reach us about anything other than support, is **stat-transfer@circlesys.com**.

Supported Programs

The following programs are supported by Stat/Transfer for Unix.

- 1-2-3
- ASCII - Delimited
- dBASE and compatible formats
- Epi Info
- Excel
- FoxPro
- Gauss
- HTML tables
- JMP
- LIMDEP
- Matlab
- Mineset
- OSIRIS (read-only)
- Quattro Pro
- SAS for Windows and OS/2
- SAS for Unix
- SAS Transport
- S-PLUS
- SPSS Data
- SPSS Portable
- Stata
- SYSTAT

The following sections give brief descriptions of each type of file including information on reading and writing each file type and how missing values are handled.

A table gives the default behavior for a given file format as input, showing the target output variable types selected by Stat/Transfer for each input variable type. A second table gives the default behavior when the file format is selected for an output file, showing the actual output types that result for each target type assigned during a transfer from some other format.

Note that all numerical variables are stored internally in Stat/Transfer as double precision floating point numbers and are then written out according to the assigned variable type.

1-2-3 Worksheet Files

Stat/Transfer will read and write files from any version of Lotus 1-2-3 (including Release 3.x and Windows).

Standard Extensions:

Lotus 1-2-3 Version 1 through 2.x	**WK1**
Lotus 1-2-3 Version 3.x and Windows	**WK3**

Reading 1-2-3 Worksheet Files

Because worksheet files are in general not designed to hold statistical data, only worksheets in certain formats can be read.

Worksheets must be in worksheet database format or in certain modifications of this. It is most straightforward when worksheets are in database format.

Database format

Worksheet database files are structured worksheets where each row is a single case and each column contains a variable. Data can consist of numbers (including serial date numbers), labels, or formulas.

You have various options to specify what part of an input worksheet to read and how to read variable names.

Data Range

By default, Stat/Transfer will read to the first non-blank cell and use that as the upper left quadrant of the data range. It will then read data until it encounters an entirely blank line. You can change the way blank lines are treated with the SET command WKS-NULL-ROWS. See Page 35.

You can change the default behavior by specifying a range with the SET command WKS-DATA-RANGE, either by giving a named range or by giving explicit coordinates. See Page 36 for details. When a range has been specified by either one of these methods, the entire range will be read and written to the output file, including blank rows.

Determining Variable Names

The first non-blank row of a worksheet database file usually has labels in each column that give the names of the variables. The data then begins in the next row. However, variable labels may be in different rows or not present at all.

By default, Stat/Transfer will attempt to determine whether or not the data in the first non-blank row (or the first row of a specified range) are variable names or the first row of data. It does so by looking for at least one column in which there is a string in the first row and a number in the second.

If this behavior is inappropriate for your worksheet (for example, if you have only string data), you can specify different options with the SET command WKS-NAME-ROW. You can specify that variable names must be taken from the first non-blank row, that they be taken from a specific row or that the worksheet does not contain variable names, so that Stat/Transfer should assign them ('col1' through 'col*n*'. See Page 36 for details.

Determining the Data Types and Widths

After identifying the label row, Stat/Transfer will look, if necessary, at the entire column to find the first non-blank cell in order to determine the data type of each column. If the first non-empty data cell of a particular column is a number (or a label with a single period), Stat/Transfer will transfer the column as a number. If the data cell contains a label, the variable will be transferred as a string.

The width of the column for each numeric variable and the format of the first non-blank data cell in that column are used, where possible, to set the default target, or output, types for the numeric variables. If the format of the first data row has any decimal places (for example, F(2)), the target type will be 'float'. On the other hand, if the cell format has no decimal places (for example, F(0)) the target types will be various flavors of integers which depend on the column width. If the column width is less than three, the target type will be 'byte'. If the column width is less than five, the target type will by 'int'. Otherwise the target type will be 'long'. Any date format in the first data row will set the target type to 'date'.

The maximum width of character variables is determined by examining the widths of all of the strings in a column.

Stat/Transfer is lenient in typing variables from worksheets. If it is expecting a character variable and it encounters a number it will convert it to a string.

Writing 1-2-3 Worksheet Files

On output, Stat/Transfer will write variable labels in the first row of the worksheet. Data values will be placed in the second and succeeding rows.

Column widths and formats will be determined by the variable information available. Dates and character variables are straightforward. For numerical data, information on the width and number of decimal places of variables, where available, is used to set the column widths and formats.

Missing Data

On input, blank cells in worksheets will be interpreted as missing values.

When transferring data to worksheets from other formats, missing values will be written out to worksheets as blank cells.

Lotus 1-2-3 Input to Stat/Transfer

Contents of First Data Row	Target Type
Number (no format)	float
Label (only a period)	float
Label (any other contents)	string
Date format Time format DateTime format	date time date-time
Numeric format in cell One or more decimal places Zero decimal places column width < 3 column width < 5 else	 float byte int long

Output to Lotus 1-2-3 from Stat/Transfer

Target Type	Output Type
byte int long float double	Numeric cell (formatted if information is available)
string	Label
date time date/time	Serial date number (with date format) Time fraction (with date format if available) Date number + time fraction (with date/time format if available)

Delimited ASCII Files

Stat/Transfer will read and write delimited ASCII.

Standard extension: **TXT**, **CSV**

Reading ASCII Delimited Files

When reading ASCII delimited files, you can specify:

- What character is read as the delimiter. You can explicitly specify the character or you can allow the program to sense it automatically.
- Whether or not the first row is treated as variable names, or whether Stat/Transfer should automatically sense it.
- The character that is used to enclose string fields in the input data.
- The maximum number of fields in the data set. The default is 256.
- The maximum width of input lines. The default is 4096 characters.
- The number of lines Stat/Transfer will read to determine each type of variable present. By default the entire data set is read.
- What character is read as missing value.
- The decimal point and thousands separator in your data.
- A "scan" format to determine whether a given field is a date, time or a date/time. The formats for dates, times and date/times are then used to actually read the data. The default formats will usually read these fields, but formats for unusual fields can be specified.
- A century changeover year. If you are reading two-digit years, you can use this option to control how they are read. The default for the option is '20'.

All of the options available for reading ASCII files are discussed in detail on Pages 29 and 32.

Writing ASCII Files

When writing ASCII delimited files, you can specify:

- The character that will be used as the delimiter in each record: commas, tabs, spaces, or semicolons.
- The character that will be used to enclose string variables on output. It is typically a double quote.
- What character will be used for missing values.
- The date and time formats.
- Whether Stat/Transfer should write variable names in the first row of the output.

See Pages 31 and 34 for details of these options.

Missing Values

By default, missing values are indicated on input and output by one delimiter immediately following another. You may change this default behavior with the SET commands NUM-MISS-RD or NUM-MISS-WR. Note that if a blank is used as the delimiter, missing values will be hard to determine.

Delimited ASCII Input to Stat/Transfer

Input Type	Target Type
Number	float
Character	string
Date Time DateTime	date time date/time

Output to Delimited ASCII from Stat/Transfer

Target Type	Output Type
byte int long float double	Number (with a precision of up to 15 decimal places)
string	Character
date time date/time	Character (written according to the ASCII format options currently in effect)

dBASE Files and Compatibles

Stat/Transfer will read and write dBASE III+ and IV files, and those from compatible systems such as Clipper or Alpha Four. Obsolete dBASE II files can also be read.

Standard Extension: **DBF**

Reading dBASE Files

All versions of dBASE files can have indices for key fields, which are stored in separate files. Stat/Transfer ignores these indices, and treats all files sequentially.

On input, dBASE numeric data and character variables are converted in a straightforward manner. Logical variables are converted to numbers ('True' becomes '1', 'False' becomes '0'). Memo Fields cannot be converted. Deleted records are not transferred.

Writing dBASE Files

Users should be aware that dBASE files are limited to 128 variables.

dBase IV files are written by default. Use the SET command WRITE-OLD-VER to write dBASE III.

dBASE III stores numeric data in fixed length character format. It is thus not very suitable for numbers which vary widely in magnitude or which are either very large or very small.

When Stat/Transfer is transferring data from a system in which the width and number of decimal places are known, it uses that information to set the format of each field in the output dBASE files. For systems, such as SYSTAT, in which this information is not recorded in the file, Stat/Transfer sets the formats based on the target type of the variable.

Missing Data

dBASE does not directly support missing values. On input to Stat/Transfer, blanks in an dBASE file are interpreted as missing values. If a data set is being transferred to an dBASE file format, missing values in the input files are set to blank in the dBASE file. Blanks are interpreted as zero by dBASE. Many other programs, including Stat/Transfer, interpret these blanks as missing.

dBASE Input to Stat/Transfer

Input Type	Target Type
Numeric (Number or Float)	
one or more decimal places	float
zero decimal places	
width < 3	byte
width < 5	int
else	long
Character	string
Logical	byte (TRUE = 1, FALSE = 0)
Date	date
Memo	not translated

Output to dBASE from Stat/Transfer

Target Type	Output Type
byte	Number (width 4, 0 decimal places)
int	Number (width 6, 0 decimal places)
long	Number (width 11, 0 decimal places)
dBASE III float double	Number (width 16, decimal places taken from input data. If decimal places unknown, set to 4.)
dBASE IV float double	Float (width 16)
string	Character
date	Date
time date/time	Character (written according to the ASCII format options currently in effect)

Epi Info

Stat/Transfer will read and write files for Epi Info, a free statistical program developed by the Center for Disease Control. All versions through Version Six are supported.

Standard Extension: **REC**

Reading Epi Info Files

The Epi Info ***.rec*** file contains both a dictionary with enough information for the program to construct a data-entry screen for the file, and the data itself in ASCII format. Stat/Transfer will use the data-entry description field as the variable label, if it is present.

Writing Epi Info Files

Because Epi Info files are basically formatted ASCII, they are not suitable for numbers which vary widely in magnitude. Keeping in mind that the ***.rec*** file is also a data-entry template, Stat/Transfer will make its best effort to make an attractive and functional one. Labels will be used, if available for variable descriptions.

Missing Data

Epi Info uses blank fields for missing values. Stat/Transfer recognizes this when reading Epi Info files. Missing values will be written as blanks on output to Epi Info format.

Epi Info Input to Stat/Transfer

Input Type	Target Type
Numeric (Number, Integer, ID)	
one or more decimal places	float
zero decimal places	
width < 3	byte
width < 5	int
else	long
Text	string
Phone number	string
Logical	byte (TRUE = 1, FALSE = 0)
Date	date

Output to Epi Info from Stat/Transfer

Target Type	Output Type
byte	Number (width 4, 0 decimal places)
int	Number (width 6, 0 decimal places)
long	Number (width 11, 0 decimal places)
float double	Number (width 16, decimal places taken from input data. If decimal places unknown, set to 4.)
string	Text
date time date/time	Date Text (written according to the ASCII format options currently in effect)

Excel Worksheets

Stat/Transfer will read and write files from Excel. It will read all versions, but will write only Version 2.1 files, which can be directly and transparently read by any version.

Standard Extension: **XLS**

Reading Excel Worksheet Files

Because worksheet files are in general not designed to hold statistical data, only worksheets in certain formats can be read.

Worksheets must be in worksheet database format or in certain modifications of this. It is most straightforward when worksheets are in database format.

Database format

Worksheet database files are structured worksheets where each row is a single case and each column contains a variable. Data can consist of numbers (including serial date numbers), labels, or formulas.

You have various options to specify what part of an input worksheet to read and how to read variable names.

Data Range

By default, Stat/Transfer will read to the first non-blank cell and use that as the upper left quadrant of the data range. It will then read data until it encounters an entirely blank line. You can change the way blank lines are treated with the SET command WKS-NULL-ROWS. See Page 35.

You can change the default behavior by specifying a range with the SET command WKS-DATA-RANGE, either by giving a named range or by giving explicit coordinates. See Page 36 for details. When a range has been specified by either one of these methods, the entire range will be read and written to the output file, including blank rows.

Determining Variable Names

The first non-blank row of a worksheet database file usually has labels in each column that give the names of the variables. The data then begins in the next row. However, variable labels may be in different rows or not present at all.

By default, Stat/Transfer will attempt to determine whether or not the data in the first non-blank row (or the first row of a specified range) are variable names or the first row of data. It does so by looking for at least one column in which there is a string in the first row and a number in the second.

If this behavior is inappropriate for your worksheet (for example, if you have only string data), you can specify different options with the SET command WKS-NAME-ROW. You can specify that variable names must be taken from the first non-blank row, that they be taken from a specific row or that the worksheet does not contain variable names, so that Stat/Transfer should assign them ('col1' through 'col*n*'. See Page 36 for details.

Determining the Data Types and Widths

After identifying the label row, Stat/Transfer will look, if necessary, at the entire column to find the first non-blank cell in order to determine the data type of each column. If the first non-empty data cell of a particular column is a number (or a label with a single period), Stat/Transfer will transfer the column as a number. If the data cell contains a label, the variable will be transferred as a string.

The width of the column for each numeric variable and the format of the first non-blank data cell in that column are used, where possible, to set the default target, or output, types for the numeric variables. If the format of the first data row has any decimal places (for example, F(2)), the target type will be ‘float’. On the other hand, if the cell format has no decimal places (for example, F(0)) the target types will be various flavors of integers which depend on the column width. If the column width is less than three, the target type will be ‘byte’. If the column width is less than five, the target type will by ‘int’. Otherwise the target type will be ‘long’. Any date format in the first data row will set the target type to ‘date’.

The maximum width of character variables is determined by examining the widths of all of the strings in a column.

Stat/Transfer is lenient in typing variables from worksheets. If it is expecting a character variable and it encounters a number it will convert it to a string.

Writing Excel Worksheet Files

On output, Stat/Transfer will write variable labels in the first row of the worksheet. Data values will be placed in the second and succeeding rows.

Column widths and formats will be determined by the variable information available. Dates and character variables are straightforward. For numerical data, information on the width and number of decimal places of variables, where available, is used to set the column widths and formats.

Missing Data

On input, blank cells in worksheets will be interpreted as missing values.

When transferring data to worksheets from other formats, missing values will be written out to worksheets as blank cells.

Excel Input to Stat/Transfer

Contents of First Data Row	Target Type
Number (no format)	float
Label (only a period)	float
Label (any other contents)	string
Date format Time format DateTime format	date time date/time
Numeric format in cell One or more decimal places Zero decimal places column width < 3 column width < 5 else	 float byte int long

Output to Excel from Stat/Transfer

Target Type	Output Type
byte int long float double	Numeric cell (formatted if information is available)
string	Label
date time date/time	Serial date number (with date format) Time fraction (with date format if available) Date number + time fraction (with date/time format if available)

FoxPro Files

Stat/Transfer will read and write FoxPro files.

Standard Extension: **DBF**

Reading FoxPro Files

FoxPro files can have indices for key fields, which are stored in separate files. Stat/Transfer ignores these indices, and treats all files sequentially.

On input, FoxPro numeric data and character variables are converted in a straightforward manner. Logical variables are converted to numbers ('True' becomes '1', 'False' becomes '0'). Memo Fields cannot be converted. Deleted records are not transferred.

Writing Fox Pro Files

Users should be aware that FoxPro files are limited to 128 variables.

FoxPro stores numeric data in fixed length character format. It is thus not very suitable for numbers which vary widely in magnitude or which are either very large or very small.

When Stat/Transfer is transferring data from a system in which the width and number of decimal places are known, it uses that information to set the format of each field in the output FoxPro files. For systems, such as SYSTAT, in which this information is not recorded in the file, Stat/Transfer uses sets the formats based on the target type of the variable.

Missing Data

FoxPro does not directly support missing values. On input to Stat/Transfer, blanks in a FoxPro file are interpreted as missing values. If a data set is being transferred to an FoxPro file format, missing values in the input files are set to blank in the FoxPro file. Blanks are interpreted as zero by FoxPro. Many other programs, including Stat/Transfer, interpret these blanks as missing.

FoxPro Input to Stat/Transfer

Input Type	Target Type
Numeric (Number or Float)	
one or more decimal places	float
zero decimal places	
width < 3	byte
width < 5	int
else	long
Character	string
Logical	byte (TRUE = 1, FALSE = 0)
Date	date
Memo	not translated

Output to FoxPro from Stat/Transfer

Target Type	Output Type
byte	Number (width 4, 0 decimal places)
int	Number (width 6, 0 decimal places)
long	Number (width 11, 0 decimal places)
float double	Float (width 16)
string	Character
date time date/time	Date Character (written according to the ASCII format options currently in effect)

Gauss Files

Stat/Transfer will read and write Gauss data sets.

There are two Gauss formats. The first, Gauss 89, is used on PC platforms, and consists of two files: a data file with a ***.dat*** extension and a header, or dictionary file with the same name and a ***.dht*** extension. The second Gauss format, Gauss 96, is used on Unix platforms and has a single file with a ***.dat*** extension. You can use a different extension for the ***.dat*** file (which will require using the file-type tag 'gauss' when specifying the file.) However, the dictionary file of Gauss 89 must always use the extension ***.dht***.

Standard Extension: **DAT**, **DHT**

Reading Gauss Files

When you wish to transfer data from a Gauss data set, give Stat/Transfer the name of the data file (the file with the ***.dat*** extension). If Stat/Transfer can find the ***.dht*** file in the same directory, it will read the data file as a Gauss 89 file. If no ***.dht*** file is present, the data file will be read as a Gauss 96 file.

Stat/Transfer will read numeric data from integer, single precision and double precision Gauss data sets. For Gauss 89 input, character data will only be read from double precision data sets.

Writing Gauss Files

On output, Gauss 96 files will be written by default. If you wish to write a Gauss 89 file, set the SET command WRITE-OLD-VER to 'Y'. If you choose to write a Gauss 89 file, both of the Gauss files, the data file and the header file, will be written. Use the ***.dat*** extension as the output file specification (or use a different extension and the file-type tag 'gauss'.) The header file will be created as well, with a ***.dht*** extension.

Stat/Transfer writes double precision Gauss data sets.

Missing Data

Gauss supports missing values.

Gauss Input to Stat/Transfer

Input Type	Target Type
Number	float
Character	string

Output to Gauss from Stat/Transfer

Target Type	Output Type
byte int long float double	Double precision number
string	Character (8 byte maximum)
date time date/time	MM/DD/YY format (written as a character variable) Character (written according to the ASCII format options currently in effect)

HTML Tables

Stat/Transfer will write HTML tables for use in web pages.

Standard Extension: **HTM**

Writing HTML Tables

Field names will be written in bold characters as the first row of the table. Values of each field are then written down the corresponding column.

The table generated by Stat/Transfer is bracketed by <HTML> tags. Thus it can be loaded directly into your browser. However, most users will probably want to cut and paste the table into a larger HTML page. To make this process easier, the output HTML tables have reasonably short lines and continuation lines are indented.

Note that HTML output is in general only appropriate for fairly small tables. Stat/Transfer will transfer large data sets to HTML format. However, if you use a large table in a web page, many browsers will be brought to their knees when they try to read the table.

Missing Data

Missing values for any data type are written as a non-breaking space, '&nbsp;'. In any browser, this will cause a blank table cell to be displayed.

Output to HTML Tables from Stat/Transfer

Target Type	Output Type
byte int long float double	Number with a precision of up to 15 decimal places
string	Character
date time date/time	Character (written according to the ASCII format options currently in effect)

JMP Files

JMP is a general statistics package from the SAS institute that runs on both the Windows and Macintosh platforms. Stat/Transfer reads and writes files that are usable on either platform.

Standard Extension: **JMP**

Reading JMP Files

JMP allows (and encourages) variable names of up to 31 characters. These variable names can contain embedded blanks and any other character. They are thus highly likely to be truncated or altered when they are transferred to other statistical packages. Because of this, Stat/Transfer will use the JMP variable name for a variable label if it is greater than eight characters and contains an embedded blank, or if you have set the SET command NUMERIC-NAMES to the option 'Y'. See Page 28.

Writing JMP Files

Variable labels, when available from the input file, will be written to JMP output files as variable "notes."

Missing Values

JMP has a missing value for each numeric type. Stat/Transfer recognizes these on reading JMP files and will write them when writing JMP output.

JMP Input to Stat/Transfer

Input Type	Target Type
Numeric	float
Integer1	byte
Integer2	int
Integer4	long
Character	string
Date Time Date/time	date time date/time

Output to JMP from Stat/Transfer

Target Type	Output Type
float double	Numeric
byte	Integer1
int	Integer2
long	Integer4
string	Character
date time date/time	Date Time Date/time

LIMDEP Files

Stat/Transfer will read and write files for LIMDEP Version 7 for Windows. LIMDEP is an econometric software program for the analysis of limited dependent variables.

Standard Extension: **LPJ**

Reading LIMDEP Files

LIMDEP has only a single data type, consisting of double precision numbers.

Writing LIMDEP Files

Stat/Transfer enforces LIMDEP's 200 variable limit.

Character variables cannot be exported to LIMDEP, since the program does not support them.

Missing Values

LIMDEP recognizes missing values.

LIMDEP Input to Stat/Transfer

Input Type	Target Type
Number	float

Output to LIMDEP from Stat/Transfer

Target Type	Output Type
string	Not exported
byte int long float double date time datetime	Number

Matlab Files

Stat/Transfer reads and write Matlab matrices. They are supported on the following platforms:

Windows & OS/2

Unix HP/Sun/IBM

Unix DEC

Standard Extension: **MAT**

Reading Matlab Files

Stat/Transfer will automatically recognize an input file's platform of origin on input.

Matlab does not write variable names into the file. Therefore Stat/Transfer makes up a variable name for each column, 'col*n*', which consists of the string 'col' plus the number of the column.

All values in a given matrix are of a single type, usually double precision, although particularly large matrices may be written out by Matlab in integer format, when this is possible.

Strings are not supported in Matlab files.

Stat/Transfer does not read complex matrices.

Writing Matlab Files

On output, Stat/Transfer writes files that can be read by any version of Matlab.

Stat/Transfer always writes Matlab files in double precision.

Stat/Transfer does not write complex matrices.

Missing Data

Matlab supports missing values.

Matlab Input to Stat/Transfer

Input Type	Target Type
Int Long Float Double	double
Character	not supported

Output to Matlab from Stat/Transfer

Target Type	Output Type
int long float double	Double
date	Date- written out as serial day number
time	Time-written as fraction of day
date/time	Date/time-integer gives serial day number, fractional part gives time
string	Not written

Mineset Files

Stat/Transfer will read and write Mineset files.

Mineset is a data visualization package from Silicon Graphics, Inc. The published and portable data format is tab-delimited ASCII, stored in a file with a ***.data*** extension and described by a dictionary in a file with the same name and a ***.schema*** extension. You can use a different extension for the dictionary file (which will require using the file-type tag 'mineset' when specifying the file.) However, the data file must always use the extension ***.data***.

Standard extensions: **SCHEMA, DATA**

Reading Mineset files

When using Mineset data as input for a transfer, give the file with the ***.schema*** extension (the dictionary file) as the input file. Stat/Transfer will then look in the same directory for a file of the same name, but with the ***.data*** extension and will read the input data from this file.

Some of the more exotic features of Mineset files such as arrays and enumerations are not supported.

Writing Mineset files.

When specifying the output file in the COPY command, give a name for the ***.schema*** file. The data file will then be written to the same directory with a ***.data*** extension.

Missing Values

Mineset uses '?' for missing numeric values. Stat/Transfer recognizes this on input of Mineset files and writes it on output to Mineset files.

Mineset Input to Stat/Transfer

Input Type	Target Type
Enum Integer	long
Float	float
Double	double
DataString String FixedString	string
Date	date

Output to Mineset from Stat/Transfer

Target Type	Output Type
byte int long	Integer
float	Float
double	Double
string	String
time date date/time	Time Date

OSIRIS Files

OSIRIS is a general purpose statistical package written for use on IBM mainframes. It is no longer actively supported. However an enormous store of survey data are available in OSIRIS format from the Inter-University Consortium for Political and Social Research (ICPSR) at the University of Michigan. For this reason, Stat/Transfer will read, but not write OSIRIS data.

An OSIRIS data set consists of a dictionary file, with the extension ***.dict***, and a data file, with the extension ***.data***. You can use a different extension for the dictionary file (which will require using the file-type tag 'osiris' when specifying the file.) However, the data file must always use the extension ***.data*** and have the same name as the dictionary file.

Standard Extension: **DICT, DATA**

Reading OSIRIS data

When naming the input file in the COPY command, you must give the file with the ***.dict*** extension (the dictionary file) as the input file. Stat/Transfer will then look in the same directory for a file of the same name, but with the ***.data*** extension and will read the input data from this file.

Stat/Transfer supports OSIRIS Type 1 data sets, which is the variant in which data are usually distributed. The dictionary file is a binary file in which strings are represented in EBCDIC notation. The data files were originally in that format, but the ICPSR generally distributes them in ASCII format. Stat/Transfer will support either of these formats and automatically senses which format your data are in.

If you are downloading an OSIRIS dictionary for use with Stat/Transfer, you should use a binary file transfer mode.

OSIRIS has variable and value labels, which are supported by Stat/Transfer.

Missing Values

An OSIRIS dictionary allows for two numeric missing values. By default, these are recognized by Stat/Transfer. However, the default behavior can changed with the SET command USER-MISS. See Page 26.

OSIRIS Input to Stat/Transfer

Input Type	Target Type
Number	
one or more decimal places	float
zero decimal places	
width < 3	byte
width < 5	int
else	long
String	string

Quattro Pro Worksheet Files

Stat/Transfer will read and write Quattro Pro files.

Standard Extension: **WQ***, **WB***

Reading Quattro Worksheet Files

Because worksheet files are in general not designed to hold statistical data, only worksheets in certain formats can be read.

Worksheets must be in worksheet database format or in certain modifications of this. It is most straightforward when worksheets are in database format.

Database format

Worksheet database files are structured worksheets where each row is a single case and each column contains a variable. Data can consist of numbers (including serial date numbers), labels, or formulas.

You have various options to specify what part of an input worksheet to read and how to read variable names.

Data Range

By default, Stat/Transfer will read to the first non-blank cell and use that as the upper left quadrant of the data range. It will then read data until it encounters an entirely blank line. You can change the way blank lines are treated with the SET command WKS-NULL-ROWS. See Page 35.

You can change the default behavior by specifying a range with the SET command WKS-DATA-RANGE, either by giving a named range or by giving explicit coordinates. See Page 36 for details. When a range has been specified by either one of these methods, the entire range will be read and written to the output file, including blank rows.

Determining Variable Names

The first non-blank row of a worksheet database file usually has labels in each column that give the names of the variables. The data then begins in the next row. However, variable labels may be in different rows or not present at all.

By default, Stat/Transfer will attempt to determine whether or not the data in the first non-blank row (or the first row of a specified range) are variable names or the first row of data. It does so by looking for at least one column in which there is a string in the first row and a number in the second.

If this behavior is inappropriate for your worksheet (for example, if you have only string data), you can specify different options with the SET command WKS-NAME-ROW. You can specify that variable names must be taken from the first non-blank row, that they be taken from a specific row or that the worksheet does not contain variable names, so that Stat/Transfer should assign them ('col1' through 'col*n*'. See Page 36 for details.

Determining the Data Types and Widths

After identifying the label row, Stat/Transfer will look, if necessary, at the entire column to find the first non-blank cell in order to determine the data type of each column. If the first non-empty data cell of a particular column is a number (or a label with a single period), Stat/Transfer will transfer the column as a number. If the data cell contains a label, the variable will be transferred as a string.

The width of the column for each numeric variable and the format of the first non-blank data cell in that column are used, where possible, to set the default target, or output, types for the numeric variables. If the format of the first data row has any decimal places (for example, F(2)), the target type will be 'float'. On the other hand, if the cell format has no decimal places (for example, F(0)) the target types will be various flavors of integers which depend on the column width. If the column width is less than three, the target type will be 'byte'. If the column width is less than five, the target type will by 'int'. Otherwise the target type will be 'long'. Any date format in the first data row will set the target type to 'date'.

The maximum width of character variables is determined by examining the widths of all of the strings in a column.

Stat/Transfer is lenient in typing variables from worksheets. If it is expecting a character variable and it encounters a number it will convert it to a string.

Writing Quattro Worksheet Files

On output, Stat/Transfer will write variable labels in the first row of the worksheet. Data values will be placed in the second and succeeding rows.

Column widths and formats will be determined by the variable information available. Dates and character variables are straightforward. For numerical data, information on the width and number of decimal places of variables, where available, is used to set the column widths and formats.

Missing Data

On input, blank cells in worksheets will be interpreted as missing values.

When transferring data to worksheets from other formats, missing values will be written out to worksheets as blank cells.

Quattro Pro Input to Stat/Transfer

Contents of First Data Row	Target Type
Number (no format)	float
Label (only a period)	float
Label (any other contents)	string
Date format Time format DateTime format	date time date-time
Numeric format in cell One or more decimal places Zero decimal places column width < 3 column width < 5 else	 float byte int long

Output to Quattro Pro from Stat/Transfer

Target Type	Output Type
byte int long float double	Numeric cell (formatted if information is available)
string	Label
date time date/time	Serial date number (with date format) Time fraction (with date format if available) Date number + time fraction (with date/time format if available)

SAS Data Files

Stat/Transfer will read and write SAS Versions 6, 7 and 8 data files for the Windows & OS/2 and Unix HP/Sun/IBM platforms.

For Unix DEC, the current version of Stat/Transfer will both read and write Version 6 files, but will only read Version 7 and 8 files.

Standard extensions:

Windows & OS/2	
Version 6	**SD2**
Versions 7-8	**SAS7BDAT**
' Unix HP/Sun/IBM	
Version 6	**SSD01**
Version 7-8	**SAS7BDAT**
Unix DEC	
Version 6	**SSD04**
Version 7-8	**SAS7BDAT**

Reading and Writing SAS Data

Reading SAS Data Files

Stat/Transfer will read data files written by SAS Version 6.08 and above.

SAS data files differ significantly between platforms. Stat/Transfer will read files written by SAS for Windows and OS/2, and files written for Unix on HP, Sun, IBM, SGI, and DEC Alpha platforms.

Stat/Transfer will automatically recognize the type of SAS file on input.

If you are moving SAS data files between platforms, you should be sure that you use a binary file transfer method.

Variable labels are supported. However, value labels are not stored as a part of the SAS data file and must therefore be transferred according to the procedure on Page 75.

Writing SAS Data Files

On output, Stat/Transfer will let you choose one of the supported file types listed above by using the appropriate extension of file-type tag.   Note that for DEC, Stat/Transfer Version 6 will read, but will not write SAS Version 7 and 8 files.

When writing SAS data files, you should pick an output format that is appropriate for the version of SAS that will be reading the file.

Value labels can be transferred according to the procedure on Page 75.

Missing Data

SAS supports missing values. On input, all of SAS's missing values are converted to a single internal missing value in Stat/Transfer. On output to SAS, missing values are set to '.', the SAS standard missing value.

SAS Data File Input to Stat/Transfer

Input Type	Target Type
Numeric	float (unless print format is: DATE DAY DDMMYY DOW JUL MMDDYY MMYY MON QTR WORDDAT WEEK YEAR or YYN which become dates, or MMSS or TIME which become times, or DATETIME or TOD which become date/times)
Character	string

Output to SAS Data Files from Stat/Transfer

Target Type	Output Type
byte	SAS Number - width three bytes
int	SAS Number - width four bytes
long	SAS Number - width six bytes
float	SAS Number - width eight bytes
double	SAS Number - width eight bytes
string	SAS Character variable (maximum length of 200 bytes)
date time date/time	SAS Serial Date Number SAS Time (number of seconds since midnight) SAS Datetime (in seconds)

SAS Value Labels

Stat/Transfer supports the import and export of SAS value labels.

Reading SAS Value Labels

SAS, unlike other systems, stores value labels as custom "formats," in a separate "catalog" file. Like the SAS file format itself, the catalog file is undocumented. Because of this, Stat/Transfer goes through the "front door" and rather than reading directly from the catalog file, reads value labels from a new file that you create using SAS itself. This file contains the same information as the catalog file, but is in a more convenient format for Stat/Transfer to read.

Creating a File with the PROC FORMAT Statement

To create the new file for Stat/Transfer to use for reading your SAS value labels, you will need to go into SAS and run the following small program:

```
libname mylib 'path' ;
proc format library = mylib
cntlout = mylib.sas_fmts ;
run ;
```

where ***path*** is the directory that contains your input data file**.**

This procedure creates a SAS file in the directory ***path*** that has the format information for each SAS data file. The file will have the name ***sas_fmts****.ext*, where *.ext* is the same as the input file extension. The file will be found in the same directory as the input SAS data file. We recommend that you use this name and location for the file, as this is the default name used by Stat/Transfer when the SAS input file is read.

Transferring Value Labels with Data

When you carry out a transfer with the SAS data file as input, Stat/Transfer will check to see if you have set the option to read value labels from a SAS datafile, READ-SAS-FMTS. If the option is 'Y', it will look for the file created with the PROC FORMAT statement. This file will then be used to transfer your variable labels.

By default it will look for ***sas_fmts****.ext* in the same directory as the input SAS file. If you would like to use a file located somewhere else or with a different name, you can change it using the SET command READ-FMT-NAME. See Page 34.

Restrictions on Importing Value Labels

SAS catalog files not only support conventional value labels (the one-to-one mapping of a string to a single number), but also the mapping of a range of numeric values to a single string (for example, zip code mapped to state). Because this latter form has no analog in any of the packages supported by Stat/Transfer, only conventional one-to-one value labels are imported from SAS.

Writing SAS Value Labels

The separate catalog file in which SAS stores value labels, besides being undocumented, can be shared among several SAS data sets. This makes it problematic to update a SAS catalog file using an external program. When writing value labels for output SAS data files, Stat/Transfer creates a SAS program that you can run in order to have SAS update its own catalog file, rather than Stat/Transfer updating the catalog file directly.

When you carry out a transfer that specifies a SAS data file as output, Stat/Transfer will check to see if you have set the WRITE-SAS-FMTS option to 'Y'. If so, it will then write a SAS PROC program file as well as a SAS output file.

If you choose to use the default name, the program file that you create will have the name of the output SAS data file with the extension ***.sas***, *outfilename**.sas***. The file will by default be placed in the same directory as the output SAS data file.

For example, if you write a SAS data file called ***out.sd2***, by default the PROC program will be in the same directory as the output file and will be named ***out.sas***.

If you would like to write the file to a different location or use a different name, you can change the default using the SET command WRITE-FMT-NAME. See Page 34.

The program file has the PROC FORMAT and MODIFY statements necessary to create the SAS catalog file. Once the program file has been created, you can run it in SAS.

SAS Transport Files

Stat/Transfer will read and write data sets in the SAS Transport format. This is, according to the SAS Institute, the "best overall format" for interfacing with other systems because it is consistent across all host environments.

If you are downloading or uploading SAS data between computers, be sure to use an error-correcting file transfer protocol that is suitable for binary files.

Standard Extension: **XPT**

Working with Transport Files within SAS

The method for writing (and reading) transport files within SAS unfortunately varies across versions of SAS.

For release 6.and higher, users, the file can be written by any DATA or PROC step that creates SAS data sets and, similarly, it can be read by any DATA or PROC step. Most commonly, PROC COPY is used to write (or to read) transport data sets. Release 6.03 users should use the SASV5XPT engine by naming it on the LIBNAME statement that defines the libref for the transport file. If you are using Version 6.06 or higher, you can read or write transport files by using the XPORT engine. To do so, you must name the XPORT engine in the LIBNAME statement.

For example, in Version 6.06 and higher the following code will write a transport file:

```
/* read system file ‘old’ - write transport file ‘trans’ */
            libname old file-specification;
            libname trans xport file-specification;
            proc copy in=old out=trans;
            run;
```

The resulting transport file can then be used for a Stat/Transfer data transfer.

If a transport file has been produced by Stat/Transfer, it can be read in SAS with the following:

```
/* read transport file ‘trans’ - write system file ‘new’ */
            libname trans xport file-specification;
            libname new file-specification;
            proc copy in=trans out=new ;
            run;
```

Version 5 users can write and read transport files by using the XCOPY procedure, the COPY procedure with the EXPORT option, or the TRANSPORT= data set option.

For further information on how to read and write transport files, particularly using Version 5, see SAS Technical Report P-195, "Transporting SAS Files between Host Systems" and the documentation for SAS on your operating system. We also recommend, particularly if you are moving files from an IBM mainframe or a VAX, that you read the excellent paper, “An Overview of Transporting SAS files between Hosts,” on the SAS Institute web site at: http://www.sas.com/techsup/download/technote/ts271.html.

Note that you should ***not*** use PROC CPORT to write files that are to be read by Stat/Transfer. This procedure creates files in an entirely different, incompatible format.

Reading SAS Transport Files

More than one data set may be stored in a single transport file. If Stat/Transfer finds more than one data set in a file, it will allow you to select the one you want, using the parameter ‘T*membername*’.

Writing SAS Transport Files

When Stat/Transfer writes a SAS transport file, it uses the file name of the input file, without the extension, as the internal name for the data set in the output file. Stat/Transfer will write only one data set to each output file.

Missing Data

SAS supports missing values. On input, all of SAS’s missing values are converted to a single internal missing value in Stat/Transfer. On output to SAS, missing values are set to ‘.’, the SAS standard missing value.

SAS Transport File Input to Stat/Transfer

Input Type	Target Type
Numeric	float (unless print format is: DATE DAY DDMMYY DOW JUL MMDDYY MMYY MON QTR WORDDAT WEEK YEAR or YYN which become dates, or MMSS or TIME which become times, or DATETIME or TOD which become date/times)
Character	string

Output to SAS Transport Files from Stat/Transfer

Target Type	Output Type
byte	SAS Number - width three bytes
int	SAS Number - width four bytes
long	SAS Number - width six bytes
float	SAS Number - width eight bytes
double	SAS Number - width eight bytes
string	SAS Character variable (maximum length of 200 bytes)
date time date/time	SAS Serial Date Number SAS Time (number of seconds since midnight) SAS Datetime (in seconds)

S-PLUS Files

Stat/Transfer will read and write S-PLUS data sets. Files written on 64 bit machines such as the DEC Alpha are not supported.

Standard Extension: [**none**]

Reading S-PLUS files

Because the S-PLUS file format is so unstructured that it allows the user to write almost anything, including code, into it, Stat/Transfer imposes a few restrictions on input files. Specifically, your data should be in one of the following formats:

- two dimensional matrix
- S-PLUS list
- dataframe

S-PLUS writes out its data in the native format of the machine on which it is running. This means that both the byte order and the width of numbers can vary between machines. On input, Stat/Transfer will automatically sense the byte order of the machine that wrote the file.

Writing S-PLUS files

On output, Stat/Transfer writes a S-PLUS dataframe. If your input data set does not have a variable named 'rownames', Stat/Transfer will create an extra variable containing the case number, stored as an integer variable and named 'rownames'.

By default, S-PLUS files will be written in the byte order appropriate to the machine being used. If you wish to change the byte order, you can use the SET command BYTE-ORDER, described on Page 28. You can choose whether you want to write out a file with low to high byte order, appropriate for the DEC Alpha or Intel processors, or a file with high to low byte order, for such processors as SPARC, HP, SGI, or IBM. If you wish to create output for the Windows version of S-PLUS, select low to high byte order on output.

Missing Data

S-PLUS supports missing values. On input, missing values are converted to the internal missing value in Stat/Transfer. On output, missing values are converted to the value appropriate for each variable type.

S-PLUS Input to Stat/Transfer

Input Type	Target Type
Integer	long
Real	float
Double	double
Logical	int
Character	string

Output to S-PLUS from Stat/Transfer

Target Type	Output Type
byte int long	Integer
float	Real
double	Double
date time date/time	Date Character (written according to the ASCII format options currently in effect}

SPSS Data Files

Stat/Transfer will read and write SPSS data files from the following platforms:

Windows and OS/2

Unix HP/Sun/IBM

Unix DEC

Standard Extension: **SAV**

Reading SPSS Data Files

Stat/Transfer automatically recognizes a file's platform of origin on input. Stat/Transfer will automatically sense the byte order of the machine that wrote the file.

Stat/Transfer will read both compressed or uncompressed SPSS data files

Writing SPSS Data Files

By default, SPSS data files will be written in the byte order appropriate to the machine being used. If you wish to change the byte order, you can use the SET command BYTE-ORDER, described on Page 28. You can choose whether you want to write out a file with low to high byte order, appropriate for the DEC Alpha and Intel processors, or a file with high to low byte order, for such processors as SPARC, HP, SGI, or IBM. If you wish to create output for the Windows version of SPSS, select low to high byte order on output.

Stat/Transfer always writes compressed files (which on typical survey data are notably smaller).

Value and variable labels are fully supported.

Missing Data

SPSS allows for three numeric missing values. By default, these are recognized by Stat/Transfer. The default behavior can be changed with the SET command USER-MISS. See Page 26. On output to SPSS, missing values are set to the SPSS system missing value.

SPSS Data File Input to Stat/Transfer

Input Type	Target Type
Number	float
Number with date format Number with time format Number with date/time format	date time date/time
Character	string

Output to SPSS Data Files from Stat/Transfer

Target Type	Output Type
byte int long float double	Number
date time date/time	Number formatted as an SPSS date Number formatted as an SPSS time Number formatted as an SPSS date/time
string	Character

SPSS Portable Files

SPSS Portable files (previously called Export files) were designed to transfer SPSS data sets between different kinds of computers. You can use them to move your data to and from mainframe SPSS-X and SPSS for the PC or Unix platforms.

Standard Extension: **POR**

Reading SPSS Portable Files

Mainframe Portable files should be transferred between computers using an error correcting communications protocol. It is quite difficult to check these files visually for errors and certain errors may fatally affect the ability of Stat/Transfer to interpret the file.

Writing SPSS Portable Files

When Stat/Transfer writes Portable files, it does so with up to ten base-thirty digits of precision.

Missing Data

SPSS allows for three numeric missing values. By default, these are recognized by Stat/Transfer. The default behavior can be changed with the SET command USER-MISS. See Page 26.

On output to SPSS, missing values are set to the SPSS system missing value.

SPSS Portable File Input to Stat/Transfer

Input Type	Target Type
Number	
Write Format, non-zero decimal places	float
Write Format, 0 decimal places	
Write width < 3	byte
Write width < 5	int
else	long
String	string

Output to SPSS Portable Files from Stat/Transfer

Target Type	Output Type
byte int long float double	Number
string	String
date time date/time	Date Character (written according to the ASCII format options currently in effect)

Stata Files

Stat/Transfer will read and write data for any version of Stata including versions running under Windows and on the Macintosh.

Standard Extension: **DTA**

Reading Stata Files

Stat/Transfer can read data from any version of Stata. Character variables and dates are fully supported. Variable and value labels are transferred out of Stata.

Writing Stata Files

Stata holds the entire data set in memory. Stat/Transfer will therefore attempt to conserve as much space as is possible. It will, in general, transfer variables into single precision floating point format.

However, when Stat/Transfer is transferring from a format in which the width and number of decimal places are known (such as SPSS, dBASE, and worksheet formats), or when it is optimizing the output variables, it will use the available information to minimize the size of your Stata data set. You can, of course, fine-tune this process by selecting types for output variables yourself.

Any variable and value labels present in the input data set will be written to Stata files.

Dates are written to Stata's internal date format.

Missing Data

Stata supports missing values.

Stata Input to Stat/Transfer

Input Type	Target Type
Byte	byte
Int	int
Long	long
Float	float
Double	double
Character	string
Number with date format	date

Output to Stata from Stat/Transfer

Target Type	Output Type
byte	Byte
int	Int
long	Long
float	Float
double	Double
date time date/time	Stata Date Float (fractional part of a day) Double (Stata date and fractional part of a day)
string	Character

SYSTAT Files

Stat/Transfer writes double precision SYSTAT files. It will read either double or single precision SYSTAT files.

Standard Extension: **SYS**

Reading SYSTAT Files

When Stat/Transfer reads SYSTAT data sets, it processes the variable names by 1) dropping the dollar signs on character variables and 2) removing the parentheses before and after subscripts. For example, SCALE(1) becomes SCALE1.

Writing SYSTAT Files

Any variable name in the source data set which contains a left parentheses followed by a number will be transferred into a SYSTAT subscripted variable.

Users should note that the SYSTAT error message, “You are trying to read an empty file,” will occur when SYSTAT cannot find a data file. Your SYSTAT files should be in the default drive or directory.

Missing Data

SYSTAT supports missing values.

SYSTAT Input to Stat/Transfer

Input Type	Target Type
Numbers	float
Character	string

Output to SYSTAT from Stat/Transfer

Target Type	Output Type
byte int long float double	Number
string	Character
date time date/time	Date Character (written according to the ASCII format options currently in effect)

Frequently Asked Questions

General Questions

Q. How big a data set can Stat/Transfer handle.

A. There is no limit on the number of cases that can be handled by Stat/Transfer.

Q. Can I share the use of Stat/Transfer with my colleagues?

A. Not if you have a personal license. Please encourage the future development and support of Stat/Transfer by complying with our license agreement.

Q. What's the best way to get support for Stat/Transfer?

A. Send us e-mail at support@circlesys.com.

Q. It's the middle of the night before a crucial deadline and my transfer won't work. What should I do?

A. It's like airplane travel. If you can't get a direct flight between Chicago and Los Angeles, try to get one that stops in Dallas. Consider what formats your destination program will read and the formats Stat/Transfer will write. Or, if you are having trouble reading a file from another program, consider any different file formats that it is capable of saving. There is usually more than one route between your source and your destination.

Q. What is the best way to pass data to my colleagues.

A. For use in general purpose PC software, Lotus 1-2-3 Version 2 files are probably the best. They are the closest thing to a *lingua franca* of data transport. They are widely supported, well documented and simple to read and write. They also have the advantage of storing numbers in double precision. If your program can save or read a file in this format, take advantage of it.

Although many programs will read dBASE files, these files have the disadvantage of not having a defined missing value and, because they store numbers in a fixed decimal format, they are not suitable for numbers that vary widely in magnitude.

For use as a general transport format between statistical packages, SPSS Portable files are probably the best. They will preserve your value and variable labels and are portable between machine architectures. On the other hand, if your colleague has Stat/Transfer (as she should), Stata files are probably the best.

SAS Data Files

Q. Stat/Transfer won't read my SAS data file. What's wrong?

A. First, we don't support every platform. If your SAS data are not in one of the formats given on Page 73, we cannot read it, and you should use SAS to create a SAS Transport file.

Further, we cannot read data that have been written with compression or which have been encrypted. If your data have been written with either of these options, you must use SAS to copy the file to an uncompressed and unencrypted format.

Finally, if you are moving the file from another platform, make sure that you use a binary, error-correcting, file transfer protocol.

If your file is in the proper format and you still cannot read it, please check our Web site to see problems that have been fixed. If your problem is not there, call or send us e-mail for technical support. The SAS file format has not been publicly documented and there may be aspects of it that we are not supporting properly. Please let us know about any problems you are having so that we can fix them for you and others.

Q. SAS refuses to read the SAS data file created by Stat/Transfer. What should I do?

A. First, make sure that you are writing the proper kind of file for the flavor of SAS you are using and that you are transporting the file properly. Then check our Web site to see if there is a problem that has already been fixed. If you think you have discovered a new problem, please let us know about it so we can fix it. In the meantime, you can always use the Transport format to get your data into SAS.

SAS Transport Files

Q. Stat/Transfer reports a "dictionary error" or otherwise refuses to read my SAS file. What is the matter?

A. Most commonly, particularly when the file is received from others, the problem is that the file is not really a transport file, but, rather, is another kind of system file. You should examine the first part of the file, either in an editor or by simply typing it to the screen. The text "HEADER RECORD*********LIBRARY HEADER RECORD" should appear at the beginning of the file. If it does not, it is not a transport file and you should refer to the Stat/Transfer manual or SAS's documentation for information on how to create a transport file in SAS.

S-PLUS Files

Q. Stat/Transfer will not read my S-PLUS file. What is wrong?

A. Remember, some S-PLUS file have very little structure and parts of the data are only meaningful to S-PLUS. Make sure that your data are in the form of a two-dimensional matrix, a list or a dataframe.

Stata Files

Q. When I transfer a 10 digit identifier to Stata, it seems to be transferred inaccurately. What is going on?

A. In order to save memory, the default numeric type when moving data to Stata is float. However, floats can only accurately represent 8 to 9 decimal digits. While this is adequate for most data, it will not do for an 10 digit identifier.

The solution is to do one of the following:

1. Manually change the type of the id number to long or double before you transfer.
2. Use the parameter '-O' with the COPY command so that Stat/Transfer will automatically put each variable in the smallest type that will maintain its precision.

Q. Why do all of my labels and variable names come out in lower case when I transfer a file to Stata.

A. That's not a bug — it's a feature. We respect the "style" of the package to which we are transferring and packages such as S-Plus and Stata favor lower case letters. If you would like to maintain the case of your variable names and labels, set the PRESERVE-CASE option.

Worksheet Files

Q. I have some blank rows in my worksheet. Stat/Transfer stops reading the data as soon as it encounters a blank row. Is there any way to work around this?

A. The reason Stat/Transfer behaves that way is that sometimes users like to put comments or notes at the bottom of their data block. If they put at least one blank line between the data and the comments, Stat/Transfer will read their data and skip the comments with no special actions on their part.

You can chenge this behavior in either of two ways:

1. You can use the SET command WKS-NULL-ROWS to change the way null rows are read.
2. You can explicitly set a data range by using the WKS-DATA-RANGE option, Stat/Transfer will return all of the rows in the range you specify.

In either case, it assumes that you know what you are doing and will return blank rows if that is what you want.

Q. When I read my Excel spreadsheet, sometimes a whole column of numbers get transferred as an integer variable instead of as a float, or date variables do not get recognized as such. What should I do?

A. You can fix the problem in either of two ways.

1. You can simply change the target type in Stat/Transfer, using the TYPES command.

2. Stat/Transfer uses the format of the first non-blank cell in a column to determine the type of each column. Make sure that these are set correctly. For date variables, you will have to make sure that you have formatted the first cell in the column with a date format.

Q. How should I represent missing values in my worksheet?

A. In general, you should just leave missing cells blank. You can also represent numeric missing data with a period. Missing data for strings can be represented by an empty string (entered with a single quote). However, blanks work just as well as any of these alternatives.

Q. Stat/Transfer just won't read my Excel worksheet?

A. Microsoft has chosen to leave some crucial tidbits out of its file format documentation. If you are having trouble with Excel, please do two things. First, use the "Save As" menu option in Excel to save your data into Lotus 1-2-3 Version 2.x format. These files will be read by a different module within Stat/Transfer. Second, make us aware of your problem, so that we can correct Stat/Transfer.

Q. It seems to take a long time to read Quattro Pro worksheets, What's the problem?

A. Quattro Pro is stored in columnwise order. Stat/Transfer must therefore transpose the entire file before it can read any of it. If your file is large, you would do better to save it out in Lotus 1-2-3 format.

Q. I want to read the second sheet of a three dimensional worksheet file. How do I do it?

A. Stat/Transfer will try and read the first non-blank page and make some sense of it. If you file has more than one page, you can use the parameter '-T*pagenumber*' to select the appropriate page. Stat/Transfer will then read the variable names and other dictionary information from that page.