Triple-S

The Survey Interchange Format Standard

A standard for porting surveys between survey packages on various hardware and software platforms

Version 1.1 **March 1998**

Introduction

This document describes the Triple-S format for survey data and variables.

Background

The aim of the Triple-S standard is to define a means of transferring the key elements of entire surveys between different survey software packages across various hardware and software platforms.

This initial version of the Triple-S standard (version 1.0) was devised by Keith Hughes, Stephen Jenkins and Geoff Wright, and published in 1994. The impetus was a paper by Peter Wills[1]. During 1996 the same group of people met to enhance and extend the standard, based on comments from implementers and users. An interim result of these meetings was presented as a paper to the ASC (Association for Survey Computing) International Conference in 1996[2]. The preliminary specification for version 1.1 of the Triple-S standard was agreed in December 1996.

Summary

A Triple-S survey is described in two ASCII text files. One, the Definition File, contains version and general information about the survey together with definitions of the survey variables. This is used to interpret the contents of the Data file. By default the Definition File has a file extension of 'SSS' and the corresponding Data File has the same name but with the extension 'ASC'.

The format of each of the files has been designed to enable software read/write routines to be easy to implement. To further aid the development process the files are relatively simple to read by eye.

Compatibility

Triple-S version 1.1 is not a superset of version 1.0. If a version 1.1 import program also accepts version 1.0 definition files, then it must interpret them identically to a version 1.0 import program.

[1] **Data Use and Reuse**, P Wills, SGCSA, Bristol England September 1992.

[2] **The Triple-S Survey Interchange Standard: The Story So Far**, S Jenkins, ASC, London England September 1996

The Definition File

Outline

The definition file is comprised of a number of keywords, some of which have associated parameters. These describe two aspects:

a. *the file itself* in terms of version number, date and time of creation etc.

b. *the survey* in terms of the survey variables.

The following shows an outline of the contents of the definition file.

```
SSS
        VERSION 1.1
        DATE "date_text"
        TIME "time_text"
        ORIGIN "origin_text"
        USER "user_text"

        SURVEY
                TITLE "survey_title_text"

                RECORD record_id
                        VARIABLE variable_id
                                . . .
                                variable_details
                                . . .
                        END VARIABLE

                        . . .
                        VARIABLE variable_id
                                . . .
                                variable_details
                                . . .
                        END VARIABLE
                END RECORD
        END SURVEY
END SSS
```

The items shown in uppercase are keywords. Those shown in lower case or as numeric digits are keyword parameters.

Basic Formatting Rules

1. General

    a. Only characters in the ASCII range decimal 32 to decimal 255 are considered valid - all others are ignored when read.

2. Keywords

    a. The case of letters in keywords is not significant. So 'a' and 'A' are equivalent, 'b' and 'B' etc.

    b. Any number of space characters (decimal 32), including none, can appear between a keyword and the following parameter. Thus VERSION 1.1 could be written VERSION1.1 and END SSS could appear as ENDSSS

    c. Individual keywords can not contain embedded space characters, thus VER SION is considered invalid.

3. Text String Parameters

    a. The double quote character, decimal 34, is used to introduce and end text strings.

    b. To represent the double quote character *within* a text string, surround the character with braces, e.g. {"}. Alternatively use the two-digit hexadecimal ASCII equivalent (surrounded by braces), e.g. {22}.

    c. To represent characters in the ASCII range decimal 0 to decimal 31 *within* a text string, use the two-digit hexadecimal ASCII equivalent surrounded by braces. One pair of braces can surround any number of consecutive hexadecimal characters, for example to represent a CR/LF sequence within a text string specify {0D0A}.

    d. To represent a newline *instruction* within a text string use the special mnemonic {NL}.

    e. To represent the open brace character *within* a text string, outside its special uses described in rules b, c and d above, surround it with braces, thus{{}. Alternatively, use the two-digit hexadecimal ASCII equivalent surrounded by braces {7B}.

4. Other Parameters

    a. The case of letters in other parameters is not significant, so that 'a' is equivalent to 'A', 'b' to 'B' etc.

    b. Any number of space characters (decimal 32), including none, can appear between parameters and any keywords. Thus POSITION 21 TO 24 could be written as POSITION21TO24.

    c. Individual (i.e. non textstring) parameters can not contain embedded space characters, thus 1 .1 is considered invalid.

5. Comments

    a. The form <comment_text> can be used to indicate parts of the definition file that are to be ignored.

    b. A <comment_text> may be used anywhere (after the VERSION 1.1 statement) outside a text string parameter wherever a space is allowed.

c. A <comment_text> may include one or more text strings (i.e. a “"” is always interpreted as starting a text string even within a comment). But note that a text string that is started within a comment must be ended within the same comment.

d. A <comment_text> may include other nested comments of the form <...> (i.e. a “<” outside a text string is always interpreted as the start of a nested comment and a “>” is always interpreted as the end of the most recently started comment).

e. The <comment_text> form should only be used where the NOTE keyword is not appropriate (e.g. manually editing out parts of the definition file).

Formatting Recommendations

In order to improve the readability of the definition file, it is recommended that:

1. The file is organised into lines using CR, LF (decimal 13, decimal 10) combinations. These are ignored when read (according to Basic Formatting Rule 1a).

2. At most one keyword, or keyword phrase, appears on one line.

3. Lines are indented with space or tab characters to reflect the structure inherent in the file. An indent is applied after every keyword that has a corresponding END keyword.

Definition File Keywords

This section describes the syntax and function of each of the keywords and keyword phrases used to form a Triple-S definition file. The keywords are shown in the order they are expected in the file.

SSS

Mandatory. Should be the first (not ignored) characters in the file.

VERSION 1.1

Mandatory. The current version is 1.1.

NOTE "note_text**"**

Optional and can appear any number of times in the definition file wherever a keyword is expected (including between values specified in a VALUES block). The importing program does not interpret the contents of the note_text. It is recommended that the NOTE form be used where comments are to be inserted into the SSS definition file by an export program.

For example: NOTE "Data collected from 12-18th June 1997"

The following four keywords (DATE, TIME, ORIGIN and USER) can appear in any order and are all optional. Each should appear at most once.

DATE "date_text**"**

Optional between VERSION and SURVEY. The parameter should represent the date the file was created.

For example: DATE "7th July 1997"

TIME "time_text**"**

Optional between VERSION and SURVEY. The parameter should represent the time the file was created.

For example: TIME "11:15"

ORIGIN "origin_text**"**

Optional between VERSION and SURVEY. The parameter should describe the originating system (program and operating system).

For example: ORIGIN "MyProg v2, Windows 95"

USER "user_text**"**

Optional between VERSION and SURVEY. The parameter should indicate the name of the user who created the file.

For example: USER "A Smith"

STANDARD standard_option

Optional between VERSION and SURVEY. This statement is used to indicate that some aspect of the SSS definition meets a defined standard that may reduce the amount of checking required by an import program.

The only standard_option provided in version 1.1 is **NAMES**. The definition of SSS standard names is:-

consist of 1-8 characters

consist of only the characters A-Z (and a-z) and 0-9

start with a letter (A-Z or a-z)

case insensitive (i.e. upper and lower case are equivalent)

unique within the survey (e.g. Q1 and Q001 are different, q1 and Q1 are the same)

appear as the "name_text" without leading, trailing, or embedded blanks

If all variable names in the SSS definition file conform to this definition then the statement **STANDARD NAMES** should appear between the VERSION and SURVEY statements.

SURVEY

Mandatory. Introduces details of the survey.

TITLE "survey_title_text**"**

Optional, but if present appears between SURVEY and RECORD. The text should represent the survey title.

For example: TITLE "The Fitness Centre Survey"

RECORD record_id

Mandatory. One RECORD keyword appears after SURVEY (or after TITLE if present). It is used to introduce the definition of the variables. The record_id is any single character A to Z (or a to z).

For example: RECORD A

The record_id can be used in conjunction with the variable_id (see the VARIABLE keyword later) to generate a unique variable name on import.

For each variable being described there should be a block comprising:

VARIABLE variable_id

Mandatory. The variable_id is an integer number of up to four digits, in the range 1 to 9999, with or without leading zeroes. Each variable_id must be unique within a RECORD ... END RECORD block.

For example: VARIABLE 10

NAME "name_text**"**

Mandatory. The name_text should represent the name the variable had in the original survey.

For example: NAME "Q1a"

Note that if the names conform to the definition of SSS standard names then a STANDARD NAMES statement should appear before the SURVEY statement.

LABEL "label_text**"**

Mandatory. The label_text should represent the label or question text of the original variable.

For example: LABEL "First visited"

TYPE type_name

Mandatory. The type of the variable as represented by type_name which must be one of:

SINGLE	- categorical with one response allowed
MULTIPLE	- categorical with any number of responses
QUANTITY	- numeric value (integer or real)
CHARACTER	- character value
LOGICAL	- Yes/No or True/False value

For example: TYPE SINGLE

POSITION start_location **[TO** finish_location **]**

Mandatory. Describes the location of the data values within the data record. The start_location and finish_location are positive integers, which represent the character positions, with the first position in the data record being 1.

For example: POSITION 21 TO 24

The **TO** clause may be omitted if the finish_location is the same as the start_location. If specified then the finish_location must be greater than or equal to the start_location.

The POSITION statement defines the part of the data record that is allocated to holding the value of a variable. The SIZE/VALUES/SPREAD statements describe which parts of the POSITION are to be interpreted as the value, and what are the legal values of the variable. As a consequence the POSITION must define an area that is at least as large as that implied by the SIZE/VALUES/SPREAD statements.

The parts of the data record defined by the POSITION statements may appear in any order, may overlap each other, and do not have to describe the entire data record. It is recommended that import programs ignore all parts of the data record not defined by POSITION statements, including those beyond the highest location defined by a POSITION statement.

The details and keywords that follow the POSITION definition vary according to the type_name parameter:-

SINGLE:	Mandatory **VALUES** block
MULTIPLE:	Optional **SPREAD** statement
	Mandatory **VALUES** block
QUANTITY:	Mandatory **VALUES** block
CHARACTER:	Mandatory **SIZE** statement
LOGICAL:	Nothing extra

SPREAD subfields **[OF** subfield_width **]**

Optional, only used with MULTIPLE type variables. The SPREAD statement indicates that the data values are coded as a series of category values in consecutive subfields (rather than as a series of 0/1 characters).

The subfields must be a positive integer, and denotes the number of subfields within the overall field that is defined by the POSITION statement. The subfield_width is also a positive integer and denotes the width of each subfield. Therefore the POSITION must define a width of at least (subfields * subfield_width).

For example: SPREAD 5 OF 3 <5 subfields of width 3>

The **OF** clause may be omitted if the subfields exactly fill the area defined by the POSITION. In this case the subfield_width is determined by dividing the width derived from the POSITION by the number of subfields.

VALUES

Mandatory for SINGLE, MULTIPLE and QUANTITY types. The VALUES block is used to define the range of legal values, optional text labels for values (e.g. categorical codes), and an optional indicator for special values (e.g. missing data codes). The end of the VALUES block is indicated by an **END VALUES** statement.

A VALUES block consists of the following statements:

start_value **TO** finish_value **[WITH]**

Optional first or only statement that indicates an overall range of legal values for the variable. The finish_value must be equal to or greater than the start_value. The **WITH** keyword is required only if there are more lines that follow in the VALUES block.

value "value_text**" [SPECIAL]**

Optional lines that are used to give labels to specific values. If no overall range has been specified then there must be at least one value defined. If an overall range has been specified then the labelled values may lie within or outside that range.

The optional **SPECIAL** marker can be used to indicate that the value may be treated in some special way (e.g. missing data, don't know etc.).

The details of the start_value, finish_value and value depend on the TYPE specification:-

For SINGLE and MULTIPLE variables:

The start_value, finish_value and value must all be positive integers. The category value lines do not need to be in any order, or be complete. There is no upper limit to the number of category values.

```
For example:   VALUES
               <3 labelled categories>
                  1"Yes"
                  2 "No"
                  9 "Refused" SPECIAL
               END VALUES


or:            VALUES
               <unlabelled with two special categories>
                  1 TO 99 WITH
                  98 "Don't Know" SPECIAL
                  99 "Refused" SPECIAL
               END VALUES
```

For QUANTITY variables:

The start_value, finish_value and value define the valid range, and implicitly define the format and physical size of data for the variable. The valid range for a Quantity can include positive or negative values. Negative values are identified by a single leading minus sign, '-'. Positive values are identified by the absence of a sign.

For example:
```
VALUES
<integers from 1 to 100>
   -1 TO 100
END VALUES
```

or:
```
VALUES
<0 to 500 with two decimal places, plus 1 special value>
   0.00 TO 500.00 WITH
   999.99 "Don't Know" SPECIAL
END VALUES
```

The number of decimal places must be the same for all values used in the VALUES block. The number of decimal places must be identical to the number of decimal places used to represent the data in the corresponding data file.

Values in the definition file must contain at least one digit. The use of a decimal point is optional for integer values. The following table gives examples of correct and incorrect representations:

Value	
1.0	Correct
+1.0	Incorrect - 'plus' sign not allowed
-1.0	Correct
- 1.0	Incorrect - contains embedded spaces
1.	Correct
.1	Correct
-.1	Correct
-.	Incorrect - no numeric digits present

There is no upper or lower limit to the magnitude of the values that may be assigned to a quantity variable.

SIZE size_specification

Mandatory for CHARACTER type variables. Defines the maximum number of characters in the data for the variable. The size_specification must be a positive integer, there is no defined upper limit to the size_specification.

Finally, for all variable types:

END VARIABLE

Mandatory. Completes definition of the variable.

Then either the definition of another variable (introduced by VARIABLE variable_id), or:

END RECORD

Mandatory. Finishes the definition for the set of variables.

END SURVEY

Mandatory. Finishes the definition for the survey.

END SSS

Mandatory. Finishes the definition file. Anything after this will be ignored.

The Data File

Overview

The data file is comprised of fixed-length records. Each record contains the responses for each of the variables in the corresponding definition file given by one respondent. All records must be of the same length, and must be at least as long as the highest location defined in a POSITION statement.

Data is recorded in fields of fixed length and arranged in the manner defined by the POSITION statements of the variables in the definition file. The type and other definitions for the corresponding variable determine the interpretation of each field.

Basic Formatting Rules

1. Other than the record terminator (see below), only characters in the range decimal 32 to decimal 255 are considered valid - any others are considered an error when read.

2. The corresponding definition file determines the minimum length of each record. This minimum length is taken from the highest location defined in a POSITION statement. There is no maximum record length.

3. Each record is terminated by either CR/LF, LF/CR, CR or LF, where CR is the carriage return character (decimal 13) and LF is the line feed character (decimal 10). Whichever terminator is used must be employed consistently - that is the same terminator must be used throughout the file.

4. The number of records in the file determines the number of respondents. There is no maximum number of records (and hence respondents) in the file.

5. There is no specific end-of-file character. The end of the file is determined by its physical size.

Individual Data Items

The following pages describe the methods used to represent data for each type of variable. In all cases, a field comprised entirely of space characters represents missing data for that variable.

Variables of type SINGLE

Data is recorded as an integer number as described in the VALUES block. The number 0 can be used to represent missing data.

The data field length is derived from the VALUES block, and is the minimum number of characters required to represent the largest value in the block. Thus, variables with values up to 9 have a data field one character long; variables with values up to 99 have a data field length of 2, and so on. If a particular data value requires less than the maximum for the field, it should be right justified using leading space or zero characters as padding.

If the data field length from the VALUES block is less than that defined in the POSITION statement then it is assumed to be right justified within the locations defined in the POSITION. Import programs should then ignore any extra parts of the position field.

For example:-

Data value	Maximum in VALUES block	POSITION width	Data record b=space, x=ignore
7	9	21 to 21	7
7	9	21 to 22	x7
7	20	21 to 22	07 or b7
7	20	21	illegal
7	20	21 to 24	xx07 or xxb7
17	20	21 to 22	17
17	20	21 to 24	xx17
142	9999	21 to 24	0142 or b142
missing	9999	21 to 24	bbbb

Variables of type MULTIPLE
Data for a Multiple may be recorded as either one character per value (bitstring format), or as a list of values (spread format).
Bitstring format
Data is recorded with one character per category of the corresponding variable. A character ‘1’ is used to signify that a category has been selected, a character ‘0’ signifies that a category is not selected. The category value refers to the relative position of the 0/1 code in the data field: thus a category value of 9 will always refer to the code in the 9th location of the data field even if some lower category values have not been defined. An import program should ignore the locations of undefined category values.

The data field length is the highest category value in the VALUES block. If the data field length is less than the POSITION statement then it is assumed to be left justified within the locations defined by the POSITION. Import programs should then ignore any extra parts of the position field.

For example,:-

Data values	Values defined in VALUES block	POSITION width	Data record b=space, x=ignore
1	1 to 9	21 to 29	100000000
1	1, 2, 3 and 9	21 to 29	100xxxxx0
1, 3	1 to 9	21 to 29	101000000
none	1 to 9	21 to 29	000000000
2, 8	1 to 9	21 to 30	010000010x
2	1, 2, 3 and 9	21 to 24	illegal
missing	1 to 9	21 to 29	bbbbbbbbb
missing	1, 2, 3 and 9	21 to 29	bbbxxxxxb

Spread format

Data is recorded as a series of subfields each containing one category value of the variable. The category value is recorded as an integer number as described in the VALUES block. The number 0 should be used to represent subfields that are not needed

The data subfield length is the minimum number of characters required to represent the largest value in the VALUES block. Thus variables with values up to 9 have a data subfield one character long, variables with values up to 99 have a data subfield length of 2, and so on. If any particular data requires less than the maximum for the subfield, it should be right justified using leading space or zero characters as padding. Data values may be stored in any or all subfields.

If the data subfield length is less than the subfield defined in the SPREAD statement then it is assumed to be right justified within the width defined in the SPREAD. Import programs should then ignore any extra parts of the subfields.

If the total width of the subfields is less than that defined in the POSITION statement then the subfields are stored consecutively left justified within the locations defined by the POSITION. Import programs should then ignore any extra parts of the position field.

For example:-

Data values	Values defined in VALUES block	SPREAD	POSITION width	Data record b=space, x=ignore
1	1 to 9	2 of 1	21 to 22	10 or 01
1	1, 2, 3 and 9	2 of 1	21 to 22	10 or 01
1, 3	1 to 9	2 of 1	21 to 22	13
1	1 to 9	2 of 2	21 to 24	x1x0 or x0x1
none	1 to 9	2 of 1	21 to 22	00
2	1, 2, 3 and 9	2 of 1	21 to 24	20xx or 02xx
1, 42	1 to 999	2 of 3	21 to 26	001042
missing	1 to 999	2 of 3	21 to 26	bbbbbb

Variables of type QUANTITY

Data is recorded as a number with the same number of decimal places as were used in the VALUES block specification of the corresponding variable. A decimal point should always appear if one was used in the VALUES block specification.

The data field length should just accommodate the longest allowable value defined by the VALUES block specification. When calculating the physical size of data for the variable, an allowance should be made for the sign of negative values. Negative numbers are represented with a leading minus sign, '-'. No such allowance should be made for (the sign of) positive values. If a particular value can be represented in a smaller length then it is right justified in the data field and leading spaces or zeros are used as padding. For negative values the spaces should appear to the left of the '-', but leading zeros should appear to the right of the '-'.

If the data field length from the VALUES block is less than that defined in the POSITION statement then it is assumed to be right justified within the locations defined in the POSITION. Import programs should then ignore any extra parts of the position field.

For example:-

Data value	Values defined in VALUES block	POSITION width	Data record b=space, x=ignore
7	0 to 99	21 to 22	b7 or 07
7.00	0.00 to 99.99	21 to 25	b7.00 or 07.00
-7	-99 to 99	21 to 23	b-7 or -07
7	-1 to 99	21 to 22	b7 or 07
7	-1 to 99	21 to 23	xb7 or x07
-1.00	-1.00 to 99.99	21 to 26	x-1.00
17	0 to 999	21 to 22	illegal
99	0 to 50 and 99	21 to 22	99
missing	0 to 999	21 to 23	bbb

Variables of type CHARACTER

Data is recorded as the original character string.

The length of the field is simply the value defined by the SIZE parameter of the corresponding variable. Data shorter than the maximum length is left justified and padded to the maximum length with trailing spaces.

For example a character variable of: SIZE 10 and data as the word 'CHARACTER' would be recorded as: CHARACTER 10

Variables of type LOGICAL

Data is recorded such that character ‘0’ represents FALSE and character ‘1’ represents TRUE.

The length of the field is always one character. If the POSITION statement defines a width of more than one character then the rightmost character is used and all others should be ignored.

For example, a value of true would be represented as: 1

Examples

Example Triple-S Definition File

The example defines a survey with six variables (one each of types SINGLE, MULTIPLE bitstring and spread format, CHARACTER, QUANTITY and LOGICAL).

```
SSS
        VERSION 1.1

        DATE "15/JUL/1997"
        TIME "18:32"
        ORIGIN "Export 1.42"
        STANDARD NAMES

        SURVEY
                TITLE "Historic House Exit Survey"
                RECORD V

                        VARIABLE 1
                                NAME "Q1"
                                LABEL "Number of visits"
                                TYPE SINGLE
                                POSITION 1
                                VALUES
                                        1 "First visit"
                                        2 "Visited before within the year"
                                        3 "Visited before that"
                                END VALUES
                        END VARIABLE

                        VARIABLE 2
                                NAME "Q2"
                                LABEL "Attractions visited"
                                TYPE MULTIPLE
                                POSITION 2 TO 10
                                VALUES
                                        1 "Sherwood Forest"
                                        2 "Nottingham Castle"
                                        3 "{"}Friar Tuck{"} Restaurant"
                                        4 "{"}Maid Marion{"} Cafe"
                                        5 "Mining museum"
                                        9 "Other"
                                END VALUES
                        END VARIABLE

                        VARIABLE 3
                                NAME "Q3"
                                LABEL "Other attractions visited"
                                TYPE CHARACTER
                                POSITION 11 TO 40
                                SIZE 30
```

```
END VARIABLE
```

```
                    VARIABLE 4
                              NAME "Q4"
                              LABEL "Two favourite attractions visited"
                              TYPE MULTIPLE
                              POSITION 41 TO 42
                              SPREAD 2        <2 subfields of width 1>
                              NOTE  "same answer list as Q2"
                              VALUES
                                        1 "Sherwood Forest"
                                        2 "Nottingham Castle"
                                        3 "{"}Friar Tuck{"} Restaurant"
                                        4 "{"}Maid Marion{"} Cafe"
                                        5 "Mining museum"
                                        9 "Other"
                              END VALUES
                    END VARIABLE

          VARIABLE 5
                              NAME "Q5"
                              LABEL "Miles travelled"
                              TYPE QUANTITY
                              POSITION 43 TO 45
                              VALUES
                                        1 TO 499
                                        500 "500 or more"
                                        999 "Not stated" SPECIAL
                              END VALUES
                    END VARIABLE

                    VARIABLE 6
                              NAME "Q6"
                              LABEL "Enjoyed visit"
                              TYPE LOGICAL
                              POSITION 46
                    END VARIABLE

          END RECORD
     END SURVEY
END SSS
```

Example Triple-S Data File

```
2101000001Amusement Park                190121
3010000000                              209991
2100100001"Marco's" Restaurant          940580
```

Interpretation

The above data corresponds to respondent data as follows:

Respondent 1:

Number of visits:	Visited before within the year	[2]
Attractions visited:	Sherwood Forest	[1]
	"Friar Tuck" Restaurant	[3]
	Other	[9]
Other attractions:	Amusement Park	
Favourite attractions:	Sherwood Forest	[1]
	Amusement Park	[9]
Miles travelled:	12	
Enjoyed visit:	TRUE	

Respondent 2:

Number of visits:	Visited before that	[3]
Attractions visited:	Nottingham Castle	[2]
Other attractions:		
Favourite attractions:	Nottingham Castle	[2]
Miles travelled:	Not stated (999)	
Enjoyed visit:	TRUE	

Respondent 3:

Number of visits:	Visited before within the year	[2]
Attractions visited:	Sherwood Forest	[1]
"Maid Marion" Cafe	[4]	
Other	[9]	
Other attractions:	"Marco's" Restaurant	
Favourite attractions:	"Marco's" Restaurant	[9]
"Maid Marion" Cafe	[4]	
Miles travelled:	58	
Enjoyed visit:	FALSE	

Changes from version 1.0

Summary

The Triple-S version 1.1 standard is based on the version 1.0 standard, but is not a true superset. The following sections provide a summary of the main changes from version 1.0.

New statements

1. The **<...>** method of identifying comments allows parts of the definition file to be skipped.
2. The **NOTE** statement allows additional narrative text to be inserted within the definition file.
3. The **STANDARD NAMES** statement will assist importing programs in generating the names of variables.
4. The **POSITION** statement means that the location of the data values in the data file are explicit. Also it allows parts of the data records to be skipped or used for more than one variable.
5. The **SPREAD** statement allows the data for Multiple type variables to be stored as actual category values in the data file.
6. The **SPECIAL** keyword allows specified data values to be flagged as being special e.g. missing value code.

Changed statements

1. The **VALUES** block has been extended so that it can define both a range of legal data values and explicitly named codes. In this new format it is now the only way to define the data values for Single, Multiple and Quantity variable types.
2. The **SIZE** statement is now only used for Character variable types.

Obsolete statements

1. The **SIZE** statement for Single, Multiple and Quantity variable types has been replaced by the value range syntax in the **VALUES** block
2. A **VALUES** block with a list of unnumbered categories is no longer supported.