triple-s XML

The Survey Interchange Standard

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

Version 1.2

August 2002

Revision 1.2.001

Introduction

This document describes the triple-s XML 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.

The 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 for a standard was a paper by Peter Wills[1] given at the SGCSA (later the ASC) conference in 1992. 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 specification for version 1.1 of the triple-s standard was published in March 1998.

A proposal for an XML translation of the standard was put forward in 1998 and triple-s XML[3] was presented to the ASC Millennium Conference in 1999. The triple-s XML version 1.1 standard was published in February 2000. Subsequently, new members Ed Ross and Laurance Gerrard joined the group. The triple-s XML 1.2 standard was published in July 2002.

More details and news of triple-s may be found at http://www.triple-s.org. To contact members of the group email info@triple-s.org.

Summary

A triple-s survey is described in two 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 XML version 1.2 is a revision of triple-s XML version 1.1 that is designed to retain compatibility with the earlier standard. The aim is that any valid XML version 1.1 specification will also be a valid XML version 1.2 specification.

---

[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

[3] **triple-s XML: A Standard Within A Standard**, K Hughes, S Jenkins, G Wright, ASC, Edinburgh Scotland September 1999

The Definition File

Outline

The definition file is coded in XML syntax according to rules given by the associated triple-s XML DTD (Document Type Definition). The definition file contents 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.

```
<?xml version="1.0"?>

<sss version="1.2" options="option_setting">
       <date>date_text</date>
       <time>time_text</time>
       <origin>origin_text</origin>
       <user>user_text</user>

       <survey>
              <name>survey_name</name>
              <title>survey_title_text</title>

              <record ident="record_id">
                     <variable ident="variable_id" type="variable_type">
                            . . .
                            variable_details
                            . . .
                     </variable>

                     . . .
                     <variable ident="variable_id" type="variable_type">
                            . . .
                            variable_details
                            . . .
                     </variable>
              </record>
       </survey>
</sss>
```

Note that the file starts with a declaration that it consists of XML.

The rest of the file is specified in terms of elements such as `<date>` and `<time>`, some of which (such as `<survey> ... </survey>`) also encapsulate other elements and some of which (such as `<record ident="record_id">`) also include attributes.

Formatting

1. Recommendations

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

a. The file is organised into lines using CR, LF (decimal 13, decimal 10) combinations. However they should be avoided within elements that contain text (e.g. `<title>`, `<label>` or `<value>`) where their presence could affect how the text is processed. Note that the `<br/>` element is provided in these situations to indicate where new line breaks should appear within the text.

b. At most one element, or element with associated attributes, appears on one line.

c. Lines are indented with space or tab characters to reflect the structure inherent in the file. An indent is applied after every element that contains other elements.

2. Comments

Comments may be used to annotate contents or to temporarily hide sections of the file from the XML parsing mechanism. These are standard XML comments and start with the conventional XML construct of `<!--` and end with `-->`. Comments are optional and can appear any number of times in the definition file.

a. `<!--`*`comment_text`*`-->` can be used anywhere (after the initial <?xml …> declaration) to indicate parts of the definition file that are to be ignored.

b. A *`comment_text`* may include any text except two successive dash characters, `--`.

For example: `<!--Data collected from 12-18th June 2002-->`

3. Unicode

XML uses the Unicode character set. As a character set, Unicode is composed of 17 planes of up to 65,536 characters each. The first plane (plane 00) is called the Basic Multilingual Plane and contains all the characters that most survey definitions would need. In order to represent such a large potential range of characters within a file XML uses "encoded" characters.

The default encoding is UTF-8. In this the ASCII characters 00-7F are represented unchanged, but all other characters are encoded as 2 or more bytes. So if the definition file only contains ASCII 7-bit characters, which includes 0-9, a-z, & A-Z, then the default UTF-8 is suitable.

However, if the variable names or more likely any texts contain 8-bit characters then they must be encoded into UTF-8 or the actual encoding must be specified. The most common encoding is Latin-1 (or more precisely ISO-8859-1) in which case the initial XML declaration should be:-

```
<?xml version="1.0" encoding="ISO-8859-1">
```

More details and resources on Unicode are available from http://www.unicode.org.

Definition File Elements and Attributes

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

```
<sss version="sss_version"
      [ options="option_setting" ]
      [ languages="language_list" ] >
```

The <sss> element is always required and is used to encapsulate the entire specification document. It contains a mandatory attribute version and optional options and languages attributes.

The version attribute is used to indicate the version of the triple-s standard that applies to this specification. If the *sss_version* is **1.1** then only elements and attributes from the triple-s XML version 1.1 standard are used. If the *sss_version* is **1.2** then new elements and attributes from the triple-s XML version standard may be present.

The options attribute is used to indicate that some aspect of the triple-s definition meets a defined standard that may reduce the amount of checking required by an import program.

The only value that *option_setting* may have in version 1.2 is **standardnames**. The definition of triple-s standardnames 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 definition file conform to this definition then the standardnames option should be set.

For example: `<sss version="1.2" options="standardnames">`

The languages attribute is used to indicate that there are some multilingual texts within the triple-s definition and to define the language identifiers that are used for those texts. See the later section on **Multilingual Texts** for more on surveys that have multi-lingual texts.

For example: `<sss version="1.2" languages="en fr">`

<date>*date_text***</date>**

Optional. The *date_text* should represent the date the file was created.

For example: `<date>20 September 1999</date>`

<time>*time_text***</time>**

Optional. The *time_text* should represent the time the file was created.

For example: `<time>18:32</time>`

<origin>*origin_text***</origin>**

Optional. The *origin_text* should describe the originating system (program and operating system).

For example: `<origin>MyProg v3, Windows 2000</origin>`

<user>*user_text***</user>**

Optional. The *user_text* should indicate the name of the user who created the file.

For example: `<user>A Smith</user>`

<survey>

Mandatory. Introduces details of the survey itself.

<name>*survey_name***</name>**

Optional. There are no specific conventions for the *survey_name*. For those systems with no specific survey naming convention this element could be used to hold the filename. The *survey_name* is not affected by the presence of a `standardnames` option in the `sss` element.

For example: `<name>SP1025</name>`

<version>*survey_version***</version>**

Optional. The content of the *survey_version* is not defined, but some local conventions will be required if it is to be used as anything more than just descriptive.

For example: `<version>3.1</version>`

<title>*survey_title_text***</title>**

Optional. The *survey_title_text* should represent the survey title. The title may optionally include any number of `<br/>` directives to indicate new line breaks. If this is a multilingual survey then the title may also contain language-specific texts (see later section on **Multilingual Texts**).

For example: `<title>The Fitness Centre Survey<br/>First wave</title>`

<record ident="*record_id***" [href="***uri***"] >**

Mandatory. One record element starts after `<survey>` (or any survey description elements if present). It is used to introduce the definition of the variables that are held in the Data File. The *record_id* is any single character A to Z or a to z.

The optional *uri* can be used to specify an explicit location for the data file that is described by this triple-s XML specification. Note that using an `href` attribute ties the specification to the data file and may cause problems if the specification and data files are moved.

For example: `<record ident="A">`

The *record_id* can be used in conjunction with the *variable_id* (see the `<variable>` element later) to generate a unique variable name on import.

Then for each variable being described there should be a block comprising:

<variable ident="*variable_id***"**
type="*variable_type***"**
[use="*use_type***"]>**

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 the `<record> ... </record>` block.

The *variable_type* 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

The *use_type* is optional and describes the role of this variable in the specification. It must be one of.

serial	this variable contains the serial number (or other identification field) for the case. There can be at most one `serial` variable and it must be either a `character` or `quantity`. The data values must be unique and should not be missing.
weight	this variable contains a case weight. There can be at most one `weight` variable and it must be a `quantity`. The data values should be non-negative and not be missing.

For example: `<variable ident="10" type="single">`

or: `<variable ident="1" type="quantity" use="serial">`

<name>*name_text***</name>**

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

For example: `<name>Q1a</name>`

Note that if the names conform to the definition of triple-s standardnames then a standardnames directive should appear in the `options` attribute of the <`sss`> element.

<label>*label_text***</label>**

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

For example: `<label>First visited</label>`

The title may optionally include any number of `<br/>` directives to indicate new line breaks. If this is a multilingual survey then the label may also contain language-specific texts (see later section on **Multilingual Texts**).

<position start="*start_location***" [finish="***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 start="21" finish="24"/>`

The *finish_location* must be greater than or equal to the *start_location*. The `finish` attribute may be omitted if the *finish_location* is the same as the *start_location*.

The `<position>` element defines the part of the data record that is allocated to holding the value of the variable. The `<size>`, `<values>` and `<spread>` elements describe which parts of the data record are to be interpreted as the value, and what are the legal values of the variable. As a consequence the `<position>` element must define an area that is at least as large as that implied by the `<size>`, `<values>` and `<spread>` elements.

The parts of the data record defined by the `<position>` elements 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>` elements, including those beyond the highest location defined by a `<position>` element.

<filter>*filter_name***</filter>**

Optional. The *filter_name* must be the name (as defined by the <`name`> element) of a previously defined **logical** variable. The value of this logical variable determines if the current variable is available for that case.

For example: `<filter>EverVisited</filter>`

The name of the logical variable must be unique. All variable types, including logical variables, can have a `<filter>` element. However, note that variables used as a `serial` or `weight` must have no missing values, hence it is inappropriate for these to have a <`filter`> specified.

The elements that can follow the `<position>` or `<filter>` element vary according to the *variable_type*:-

single	Mandatory **values** element
multiple	Optional **spread** element
	Mandatory **values** element

quantity	Mandatory **values** element
character	Mandatory **size** element
logical	Nothing extra

<spread subfields="*num_subfields***" [width="***subfield_width***"] />**

Optional and only used with multiple type variables. The <spread> element indicates that the data values are coded as a series of category values in consecutive subfields (rather than the default multiple format of a series of 0/1 characters).

The *num_subfields* attribute must be a positive integer, and denotes the number of subfields within the overall field that is defined by the <position> element. The *subfield_width* is also a positive integer and denotes the width of each subfield. Therefore the <position> element must define a width of at least (*num_subfields* * *subfield_width*). The *subfield_width* must be large enough to hold the largest category value specified for the multiple.

For example: <spread subfields="5" width="3"/> …5 subfields of width 3

The width attribute may be omitted if the *num_subfields* exactly fills the area defined by the <position> element. In this case the *subfield_width* is determined by dividing the width derived from the <position> element by *num_subfields*.

<values> … **</values>**

Mandatory for single, multiple and quantity types. The <values> element is used to define the set of legal values and optional text labels for values (e.g. categorical codes).

A <values> element contains at least one of a <range> or one or more <value> elements. If a <range> is present then it must be the first element.

The details of the *start_value*, *finish_value* and *code_value* depend on the type specification.

<range from="*start_value***" to="***finish_value***" />**

Optional first or only element. The <range> indicates an overall range of legal values for the variable. The *finish_value* must be equal to or greater than the *start_value*. This may be followed by any number of <value> elements each defining a particular value.

<**value code**="*code_value*"**>***value_text***</value**>

Any number of optional elements that are used to give labels to specific values. The *value_text* may optionally include any number of
 directives to indicate new line breaks. If this is a multilingual survey then the value may also contain language-specific texts (see later section on **Multilingual Texts**).

If no <range> element has been specified then there must be at least one <value> element. If a <range> element has been specified then the *code_value* may lie within or outside the defined *start_value* and *finish_value*. Apart from this, all *code_values* must be unique within each <values> element.

For single and multiple variables:

The *start_value*, *finish_value* and *code_value* must all be positive integers. The <value> elements do not need to be in any order, nor be complete. There is no upper limit to the number of <value> elements, which may be specified within the corresponding variable definition.

For example:
```
<values>
<!--3 labelled categories-->
    <value code="1">Yes</value>
    <value code="2">No</value>
    <value code="9">Refused</value>
</values>
```

or:
```
<values>
<!--unlabelled with two explicit categories-->
    <range from="1" to="99" />
    <value code="98">Don't Know</value>
    <value code="99">Refused</value>
</values>
```

For quantity variables:

The *start_value*, *finish_value* and *code_value* explicitly define the valid range, and implicitly define the format and physical size of data for the variable. The valid range for a variable of type **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-->
    <range from="1" to="100" />
</values>
```

or:
```
<values>
<!—0 to 500 with 2 dp, plus 1 explicit value-->
    <range from="0.00" to="500.00" />
    <value code="999.99">Don't Know</value>
<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***</size>**

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*.

For example: `<size>100</size>`

Finally, for all variable types:

</variable>

Mandatory. Completes definition of the variable.

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

</record>

Mandatory. Finishes the definition for the set of variables.

</survey>

Mandatory. Finishes the definition for the survey.

</sss>

Mandatory. Finishes the definition file.

Multilingual Texts

The text for a `survey <title>`, `variable <label>` or `values <value>` element may contain any number of language-specific text elements. These are in addition to the plain or default text that should always be present for these elements.

For example:
```
<value code="1">Yes
      <text xml:lang="en-GB">Yes</text>
      <text xml:lang="en-US">Sure</text>
      <text xml:lang="fr">Oui</text>
</value>
```

<text xml:lang="language_identifier"**>***formatted_text***</text>**

Optional. The *formatted_text* may optionally include any number of `<br/>` directives to indicate new line breaks.

Although there is no restriction on `language_identifier`, the intended values of the `xml:lang` attribute are described in the official W3C XML version 1.0 specification as:-

"The values of the attribute are language identifiers as defined by IETF (Internet Engineering Task Force) RFC 1766, *Tags for the Identification of Languages* (http://www.ietf.org/rfc/rfc1766.txt) or its successor on the Standards Track.

Note:

IETF RFC 1766 tags are constructed from two-letter language codes as defined by ISO 639 (*Codes for the representation of names of languages*), from two-letter country codes as defined by ISO 3166 (*Codes for the representation of countries and their subdivisions – part 1 (country codes)*), or from language identifiers registered with the Internet Assigned Numbers Authority, *Register of Language Tags*. It is expected that the successor to IETF RFC 1766 will introduce three-letter language codes for languages not presently covered by ISO 639."

The use of multilanguage texts within the specification must be signalled by a `languages="list"` attribute in the initial `<sss>` element.

For example: `<sss version="1.2" languages="en-GB en-US fr">`

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>` element.

Data is recorded in fields of fixed length and arranged in the manner defined by the `<position>` elements 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 by the <values> element. The number 0 can be used to represent missing data.

The data field length is derived from the <value> and <range> elements in the <values> element, and is the minimum number of characters required to represent the largest value. 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 each <value> or <range> element is less than that defined in the corresponding <position> element then it is assumed to be right justified within the locations defined in the <position> element. Import programs should then ignore any extra.

For example:-

Data value	Maximum in <values> element	<position> element	Data record b=space, x=ignore
7	9	start="21" finish="21"	7
7	9	start="21" finish="22"	x7
7	20	start="21" finish="22"	07 or b7
7	20	start="21"	illegal
7	20	start="21" finish="24"	xx07 or xxb7
17	20	start="21" finish="22"	17
17	20	start="21" finish="24"	xx17
142	9999	start="21" finish="24"	0142 or b142
missing	9999	start="21" finish="24"	bbbb

Variables of type multiple

Data for Multiples 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 associated value or range elements. If the data field length is less than the position element 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 value	Maximum in <values> element	<position> element	Data record b=space, x=ignore
1	1 to 9	start="21" finish="29"	100000000
1	1, 2, 3 and 9	start="21" finish="29"	100xxxxx0
1, 3	1 to 9	start="21" finish="29"	101000000
none	1 to 9	start="21" finish="29"	000000000
2, 8	1 to 9	start="21" finish="30"	010000010x
2	1, 2, 3 and 9	start="21" finish="24"	illegal
missing	1 to 9	start="21" finish="29"	bbbbbbbbb
missing	1, 2, 3 and 9	start="21" finish="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 element. 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 element 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 element 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 value	Maximum in <values> element	<spread> element	<position> element	Data record b=space, x=ignore
1	1 to 9	subfields="2" width="1"	start="21" finish="22"	10 or 01
1	1, 2, 3 and 9	subfields="2" width="1"	start="21" finish="22"	10 or 01
1, 3	1 to 9	subfields="2" width="1"	start="21" finish="22"	13
1	1 to 9	subfields="2" width="2"	start="21" finish="24"	x1x0 or x0x1
none	1 to 9	subfields="2" width="1"	start="21" finish="22"	00
2	1, 2, 3 and 9	subfields="2" width="1"	start="21" finish="24"	20xx or 02xx
1, 42	1 to 999	subfields="2" width="3"	start="21" finish="26"	001042
missing	1 to 999	subfields="2" width="3"	start="21" finish="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 element specification of the corresponding variable. A decimal point should always appear if one was used in the values element specification.

The data field length should just accommodate the longest allowable value defined by the values element 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 element is less than that defined in the position element 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	<range> element	<position> element	Data record b=space, x=ignore
7	from="0" to="99"	start="21" finish="22"	b7 or 07
7.00	from="0.00" to="99.99"	start="21" finish="25"	b7.00 or 07.00
-7	from="-99" to="99"	start="21" finish="23"	b-7 or -07
7	from="-1" to="99"	start="21" finish="22"	b7 or 07
7	from="-1" to="99"	start="21" finish="23"	xb7 or x07
-1.00	from="-1.00" to="99.99"	start="21" finish="26"	x-1.00
17	from="0" to="999"	start="21" finish="22"	illegal
99	from="0" to="50" with additional <value code="99">	start="21" finish="22"	99
missing	from="0" to="999"	start="21" finish="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 element of the corresponding variable. If the data field length from the size element is less than that defined in the position element then it is assumed to be left justified within the locations defined in the position. Import programs should then ignore any extra parts of the position field.

For example a character variable of: `<size>10</size>` and data as the word `character` would be recorded as: `"character "`

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>` element 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).

```
<?xml version="1.0" encoding="ISO-8859-1">
<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v1.2//EN"
                     "http://www.triple-s.org/dtd/sss_v12.dtd">

<sss version="1.2" options="standardnames">

<date>21 September 2001</date>
<time>18:32</time>
<origin>Export 1.42</origin>

<survey>
    <name>SP5201-1</name>
    <title>Historic House Exit Survey<br/>First Wave</title>

    <record ident="V">

        <variable ident="1" type="single">
            <name>Q1</name>
            <label>Number of visits</label>
            <position start="1" />
            <values>
                <value code="1">First visit</value>
                <value code="2">Visited before within the year</value>
                <value code="3">Visited before that</value>
            </values>
        </variable>

        <variable ident="2" type="multiple">
            <name>Q2</name>
            <label>Attractions visited</label>
            <position start="2" finish="10" />
            <values>
                <value code="1">Sherwood Forest</value>
                <value code="2">Nottingham Castle</value>
                <value code="3">&quot;Friar Tuck&quot; Restaurant</value>
                <value code="4">&quot;Maid Marion&quot; Cafe</value>
                <value code="5">Mining museum</value>
                <value code="9">Other</value>
            </values>
        </variable>

        <variable ident="3" type="character">
            <name>Q3</name>
            <label>Other attractions visited</label>
            <position start="11" finish="40" />
            <size>30</size>
        </variable>
```

```
        <variable ident="4" type="multiple">
            <name>Q4</name>
            <label>Two favourite attractions visited</label>
            <position start="41" finish="42" />
            <spread subfields="2" />
            <!--same answer list as Q2-->
            <values>
                <value code="1">Sherwood Forest</value>
                <value code="2">Nottingham Castle</value>
                <value code="3">&amp;quot;Friar Tuck&amp;quot; Restaurant</value>
                <value code="4">&amp;quot;Maid Marion&amp;quot; Cafe</value>
                <value code="5">Mining museum</value>
                <value code="9">Other</value>
            </values>
        </variable>

        <variable ident="5" type="quantity">
            <name>Q5</name>
            <label>Miles travelled</label>
            <position start="43" finish="45" />
            <values>
                <range from="1" to="499" />
                <value code="500">500 or more</value>
                <value code="999">Not stated</value>
            </values>
        </variable>

        <variable ident="6" type="logical">
            <name>Q6</name>
            <label>Would come again</label>
            <position start="46" />
        </variable>

        <variable ident="7" type="single">
            <name>Q7</name>
            <label>When is that most likely to be</label>
            <position start="47" />
            <filter>Q6</filter>
            <values>
                <value code="1">Within 3 months</value>
                <value code="2">Between 3 months and 1 year</value>
                <value code="3">More than 1 years time</value>
            </values>
        </variable>

        <variable ident="99" type="quantity">
            <name>Q99</name>
            <label>Case weight</label>
            <position start="48" finish="54"/>
            <values>
                <range from="0.0000" to="99.9999"/>
            </values>
        </variable>

    </record>

</survey>

</sss>
```

Example triple-s Data File

```
2101000001Amusement Park                190121101.4000
3010000000                              209991300.9000
2100100001"Marco's" Restaurant          940580 00.7000
```

Interpretation

The pervious 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	
Would visit again:	TRUE	
Next visit:	Within 3 months	[1]
Case weight	1.4000	

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)	
Would visit again:	TRUE	
Next visit	More than a years time	[3]
Case weight	0.9000	

Respondent 3:

Number of visits:	Visited before within the year	[2]
Attractions visited:	Sherwood Forest	[1]
	"Maid Marion" Café	[4]
	Other	[9]
Other attractions:	"Marco's" Restaurant	
Favourite attractions:	"Marco's" Restaurant	[9]
	"Maid Marion" Café	[4]
Miles travelled:	58	
Would visit again:	FALSE	
Next visit:	-- not applicable --	
Case weight	0.7000	

The triple-s XML DTD

Summary

The triple-s XML 1.2 DTD is given below. As with all XML code, this document is required if the syntax of a triple-s XML Description File is to be verified as 'valid' rather than simply being considered 'well formed'. Note that this DTD is available online from http://www.triple-s.org/dtd/sss_v12.dtd.

```
<!-- triple-s.dtd                                                    -->
<!-- Standalone XML version 1.12                                     -->
<!-- June 2002                                                       -->
<!-- ============================================================    -->
<!-- An XML definition for moving surveys between packages           -->
<!-- on various hardware and software platforms.                     -->
<!-- ============================================================    -->
<!-- This DTD has been produced by the triple-s group                -->
<!-- (Keith Hughes, Steve Jenkins, Geoff Wright,                     -->
<!--  Ed Ross and Laurance Gerrard).                                 -->
<!-- For further information on the triple-s group visit             -->
<!-- the web site at http://www.triple-s.org.                        -->
<!-- ============================================================    -->
<!-- Public identifier:                                              -->
<!--   -//triple-s//DTD Survey Interchange v1.2//EN                  -->
<!-- Public URL:                                                     -->
<!--   http://www.triple-s.org/dtd/sss_v12.dtd                       -->
<!-- ============================================================    -->
<!--                                                                 -->
<!-- Version history:                                                -->
<!--                                                                 -->
<!--   1.0             Original non-XML triple-s standard            -->
<!--   1.1   Feb 2000  XML version based on non-XML 1.1 standard     -->
<!--   1.1a  Apr 2001  type attribute (variable element) lower case  -->
<!--                   code attribute (value element) REQUIRED       -->
<!---                  size element definition moved within DTD      -->
<!--   1.1b  Jun 2001  added formatted texts (with break element)    -->
<!--                     for title, label and value elements         -->
<!--   1.2   Jun 2002  XML version 1.2 (initial version)             -->
<!--                                                                 -->
<!-- ============================================================    -->
<!--                                                                 -->
<!--          BEGINNING OF ACTUAL DOCUMENT TYPE DEFINITION           -->
<!--                                                                 -->

<!--temporary parameter entities                                     -->
<!ENTITY % vartype "single |
                    multiple |
                    quantity |
                    character |
                    logical" >
<!ENTITY % usetype "serial |
                    weight" >

<!-- formatted text with breaks element and entity                  -->
<!ELEMENT br EMPTY>
<!ENTITY % formatted_text "(#PCDATA | br)*" >

<!-- multi-lingual texts                                            -->
<!ELEMENT text (%formatted_text;)*>
<!ATTLIST text xml:lang NMTOKEN #REQUIRED>
<!ENTITY % texts "(%formatted_text; | text)*">

<!-- TOP LEVEL                                                      -->
<!-- =========                                                      -->
<!--                                                               -->
<!ELEMENT sss (date?, time?, origin?, user?, survey)>
<!ATTLIST sss version CDATA #REQUIRED
              options NMTOKENS #IMPLIED
              languages NMTOKENS #IMPLIED>
```

```
<!-- DOCUMENT DESCRIPTION                                              -->
<!-- ====================                                              -->
<!--                                                                   -->
    <!ELEMENT date (#PCDATA)>

    <!ELEMENT time (#PCDATA)>

    <!ELEMENT origin (#PCDATA)>

    <!ELEMENT user (#PCDATA)>

<!--                                                                   -->
<!-- SURVEY DESCRIPTION                                                -->
<!-- ==================                                                -->
<!--                                                                   -->
    <!ELEMENT survey (name?, version?, title?, record)>

        <!ELEMENT name (#PCDATA)>

        <!ELEMENT version (#PCDATA)>

        <!ELEMENT title %texts;>

<!--                                                                   -->
<!-- RECORD DESCRIPTION                                                -->
<!-- ==================                                                -->
<!--                                                                   -->
        <!ELEMENT record (variable+)>
        <!ATTLIST record ident CDATA #REQUIRED
                         href CDATA #IMPLIED>

<!--                                                                   -->
<!-- VARIABLE DESCRIPTION                                              -->
<!-- ====================                                              -->
<!--                                                                   -->
            <!ELEMENT variable (name, label, position, filter?,
                    ((spread?, values) | size)?)>
            <!ATTLIST variable ident CDATA #REQUIRED
                               type (%vartype;) #REQUIRED
                               use (%usetype;) #IMPLIED>>

                <!ELEMENT name (#PCDATA)>

                <!ELEMENT label %texts;>

                <!ELEMENT position EMPTY>
                <!ATTLIST position start CDATA #REQUIRED
                                   finish CDATA #IMPLIED>

                <!ELEMENT filter (#PCDATA)>

                <!ELEMENT spread EMPTY>
                <!ATTLIST spread subfields CDATA #REQUIRED
                                 width CDATA #IMPLIED>

                <!ELEMENT size (#PCDATA)>

<!--                                                                   -->
<!-- VALUES DESCRIPTION                                                -->
<!-- ==================                                                -->
<!--                                                                   -->

                <!ELEMENT values (value+ | (range, value*))>

                    <!ELEMENT value %texts;>
                    <!ATTLIST value code CDATA #REQUIRED>

                    <!ELEMENT range EMPTY>
                    <!ATTLIST range from CDATA #REQUIRED
                                    to CDATA #REQUIRED>

<!--                                                                   -->
<!-- END OF DOCUMENT TYPE DEFINITION                                   -->
```

Changes from triple-s XML 1.1 to triple-s XML 1.2

Summary

The triple-s XML version 1.2 standard is based on the XML version 1.1 standard, and should be a true superset (i.e. all triple-s XML 1.1 specifications should be valid within triple-s XML 1.2) The following sections provide a summary of the main additions from XML version 1.1.

1. The survey NAME and VERSION elements to give more information about the survey itself.

2. The FILTER element to specify simple routing and filtering preconditions for individual variables.

3. The USE attribute on individual variables allows a SERIAL and/or WEIGHT variable to be specified for the survey.

4. The TEXT element allows the survey <title>, variable <label> and values <value> elements to include any number of language-specific texts.

5. The HREF attribute on the RECORD element means that the location of the data file can be defined explicitly.

Changes from triple-s 1.1 to triple-s XML 1.1

Summary

The definition file is now expressed in XML syntax according to the rules expressed in the triple-s XML 1.1 DTD. triple-s XML version 1.1 implements the same feature set as triple-s version 1.1 with the exceptions:-

1. The SPECIAL directive (for a VALUE) is now obsolete.

2. The NOTE statement has been replaced by standard XML comments.

3. The `<br/>` element as part of the `<title>`, `<label>` or `<value>` elements supersedes the use of the {NL} (new line) directive.

Changes from triple-s 1.0 to triple-s 1.1

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 notes to be inserted within the definition file.

3. The **standardnames** option will assist importing programs in generating the names of variables.

4. The specifications for the **position** statement mean that the location of the data values in the data file are explicit. Parts of the data record may 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.

Changed statements

1. The **values** statement can (now) define both a range of legal data values and explicitly named codes. In this new specification 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.