Triple-S XML

The Survey Interchange Standard

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

Version 2.0 **September 2011**

Revision 2.0.002

Published by:

ASC

PO Box 76, Berkeley, Gloucestershire, UK, GL13 9WU

Tel: +44 (0) 1453 511511

Fax: +44 (0) 1453 511512

Email: admin@asc.org.uk

Website: www.asc.org.uk

On behalf of the Triple-S Group:

Laurance Gerrard

Keith Hughes

Steve Jenkins

Ed Ross

Geoff Wright

Email: info@triple-s.org

Website: www.triple-s,.org

ISBN: 0-9546748-3-9

Table of Contents

Foreword

Foreword to the 2011 Edition

So often, we have a perception that things change so rapidly in IT that anything created a year or two ago is already out-of-date. Therefore, to actively support something that has changed little if at all in the previous five years may seem lazy or foolish. On the contrary, Triple-S is a standard that anyone involved in market research software would be foolish to ignore. Five years on from the release of version 2.0, the Triple-S group is releasing a new version of the manual with a small number of improvements that its users have been asking for.

There are good reasons why the team that support this remarkable standard for market research data take a cautious approach to updates. We are all only too aware of the conflicts that can arise in exchanging documents between the many different versions of Microsoft Office, to provide just one example. Here, change has been frequent and subsequent versions have introduced incompatibilities which have left users either cut off from their data, or marooned on an island of their own, using a version of software that others have abandoned.

With standards, on the other hand, change is best achieved slowly, with each new version requiring time to mature through adoption by those that have implemented it. Ironically, by moving slowly, this allows developers to push forward with their own much more frequent releases of their software, if they support Triple-S as a data exchange mechanism.

There are now over seventy users who have registered with Triple-S as adopters, but I am aware of many others – the real figure probably now exceeds a hundred. The standard may have started life as a standard used principally in the United Kingdom, but that has long ceased to be the case. This is a truly international standard, adopted by software developers across the world.

If you are someone who benefits from using Triple-S as an implementer, the best way to express your appreciation to the Triple-S group is to register on the Triple-S website. This helps everyone to understand the extent to which this extraordinary initiative now underpins such a large proportion of professional research conducted day by day – and this will also go a long way to secure its continued success.

Tim Macer
London, August 2011

Foreword to the 2005 Edition

By happy coincidence, the publication of Triple-S 2.0 comes at a time when the number of software developers supporting one or more prior versions of the standard has edged passed the very creditable 50 mark. Triple-S has moved from being a standard unknown outside of the UK, where it originated, to one which developers in the USA, Australia, across Europe and further afield, are building into their products, and one which users around the globe now rely on every day.

It has been a slow burn since three software manufacturers got together in 1994 to see if a shared standard could put an end to customers having to waste effort reformatting data between almost antagonistically incompatible software packages. Enlightened self-help was certainly one motivation, as the standard has also spared these developers from constantly re-writing their own import and export tools to satisfy an ever-changing line-up of software products. In a dozen years, Triple-S has changed twice and the number of exports *not* written as a result must now be in the thousands. Given the hazards of measuring an absence, it is still probably fair to say that the existence of Triple-S has saved software developers, both the original three companies and to a much larger extent their peers and rivals, at least $1.5 million (around €1.4 m) in development costs since its inception. The savings to the marketing research and opinion survey industry must be well in excess of this every year.

Over the years, some critics have considered the standard too limited in scope. But the Triple-S group have rightly, in my view, taken a cautious approach to upgrading the standard for version 2.0 and focused on universal demands. For example, hierarchical datasets were not previously represented in Triple-S, but are common in marketing research and are notoriously difficult to handle without specific support. They now form a welcome and elegant addition. The standard was also a little dogmatic in some places about how certain data values should be represented, but was silent on other widely used values such as dates, times or semantic scale score values. These too are addressed for the first time in version 2.0.

For some critics these and the other changes described in this revised standard will still be too modest, as they are looking for Triple-S to model much more of the complexity that there is in research data collection, or the different ways in which each solution on the market or developed in-house represents this. It is not something the standard can or should do. A common ground approach is vital. The success of Triple-S has hinged on occupying this middle ground. It offers much more than a lowest common denominator – which in this case is probably the painfully inadequate CSV file – but stops well short of forcing every adopter to wrestle with arcane and obscure requirements demanded by very few. A complex standard would simply take too long for each software developer to implement or to upgrade to, when a new version is published, as it is now.

Furthermore, by being modelled on a consensual view of marketing and opinion research data, the standard is truly a servant of the industry and is not a hostage to the proprietary (and also limited) view of one or two widely used statistical packages that are often misleadingly presented as a *de facto* standard for data transfer.

Perhaps the greatest virtue of Triple-S is what excellent value for money it represents. For developers, it is free to implement, for software users, it is effectively free to use and yet for both groups of ‘customers’ it inevitably saves time and money. It is only free because of the generosity of the individual members of the Triple-S group who freely give of their time, knowledge and experience in developing and supporting it, and also the small number of organisations who financially support their vital work. I salute each one of them for their foresight, and congratulate them on reaching this important milestone.

If you are referring to this revision of the standard with a view to implementing it or using it for the first time, or upgrading your current implementation, I congratulate you too and hope that you will find it in your heart to express your appreciation, in one form or another, to the standard’s authors.

Tim Macer
London, May 2006

Tim Macer, Managing Director of meaning ltd, is an independent technology analyst and observer in the field of research. He writes in Research magazine in the UK and Quirk’s Marketing Review in the United States.

Introduction

This document describes version 2.0 of 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, and included some fairly limited extensions.

[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

Consideration of more significant enhancements to the standard started after a presentation on the future direction for Triple-S[4] and an open meeting at the ASC Conference in September 2003. Options for possible improvements were then discussed at a meeting with a number of developers and users in March 2004. This resulted in a reduced set of changes which were then refined during the rest of 2004. The resulting Triple-S XML 2.0 standard was published in April 2005. An online Validator for Triple-S metadata files was published in 2010 and is available from the Triple-S website.

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 Metadata 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. It is recommended that the Metadata File has a file extension of ‘xml’ (or 'sss' for compatibility with previous versions of the standard) and the corresponding Data File has the same name but with the extension 'asc', or if comma-delimited data is used, the extension ‘csv’.

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.

A further development in Triple-S 2.0 is the addition of an optional Hierarchical Control File. This XML file can be used to describe how a series of simple Triple-S Metadata and Data Files can be combined to represent a hierarchical survey.

Compatibility

Triple-S XML version 2.0 is developed from Triple-S XML version 1.2. It is designed to retain substantial compatibility with the earlier standard. The aim is that any valid Triple-S XML version 1.2 specification should require only limited, or even no, changes to become a valid XML version 2.0 specification.

[4] **triple-s: Managing Success**, G Wright, ASC, Warwick England September 2003

The Metadata File

Outline

The Metadata File is coded in XML syntax according to rules given by the associated Triple-S XML DTD (Document Type Definition). The Metadata 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, or the *hierarchy* in terms of the contributing Metadata Files.

The following shows an outline of the contents of the Metadata File for a survey.

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

<sss version="2.0">
       <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_ident">
                     <variable ident="variable_ident"
                     type="variable_type">
                            . . .
                            variable_details
                            . . .
                     </variable>

                     . . .
                     <variable ident="variable_ident"
                     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_ident`*`">`) also include attributes.

XML

Introduction

Extensible Markup Language (XML) is a set of rules for encoding documents in a machine-readable form. It is defined in the XML 1.0 Specification[5] produced by the W3C, and several other related specifications, all open standards.

The design goals of XML emphasise simplicity, generality, and usability. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is also widely used for the representation of arbitrary data structures.

The use of XML to describe the metadata has implications on the terminology used within the Triple-S standard, and the rules for specifying the Metadata File.

Terminology

The characters which make up an XML document are divided into *markup* and *content*. Markup and content may be distinguished by the application of simple syntactic rules. All strings which constitute markup either begin with the character "<" and end with a ">", or begin with the character "&" and end with a ";". Strings of characters which are not markup are content.

A **Tag** is a markup construct that begins with "<" and ends with ">". Tags come in three flavours: *start-tags*, for example `<section>`, *end-tags*, for example `</section>`, and *empty-element tags*, for example `<line-break />`.

An **Element** either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. The characters between the start- and end-tags, if any, are the element's *content*, and may contain further markup,

[5] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml/

including other elements, which are called *child elements*. An example of an element is `<Greeting>Hello, world.</Greeting>.`

An **Attribute** is a markup construct consisting of a name/value pair that exists within a start-tag or empty-element tag. An example would be `<step number="3">Connect A to B.</step>` where the name of the attribute is "number" and the value is "3".

The XML specification defines a document as a text that is well-formed, i.e. it satisfies a list of syntax rules provided in the XML specification. The list is fairly lengthy; some key points are:

- It contains only properly encoded legal Unicode characters.
- None of the special syntax characters such as "<" and "&" appear except when performing their markup-delineation roles.
- The begin, end, and empty-element tags that delimit the elements are correctly nested, with none missing and none overlapping.
- The element tags are case-sensitive; the beginning and end tags must match exactly.
- There is a single "root" element that contains all the other elements.

In addition, to be a valid Triple-S Metadata File the XML document must conform to the rules of the Triple-S DTD as defined in this standard.

Unicode

XML uses the Unicode[6] character set. Unicode provides a consistent way of encoding multilingual plain text and brings order to a chaotic state of affairs that has made it difficult to exchange text files internationally.

The design of Unicode is based on the simplicity and consistency of ASCII, but goes far beyond ASCII's limited ability to encode only the Latin alphabet. The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. To keep character coding simple and

[6] The Unicode Consortium, **http://www.unicode.org**

efficient, the Unicode Standard assigns each character a unique numeric value and name.

An encoding defines how characters are to be represented in a byte, word or double word oriented format (i.e. in 8, 16 or 32-bits per code unit). Some encodings only permit a subset of the Unicode character set to be represented (e.g. 7-bit ASCII). Other encodings (e.g. UTF-8) can be used to represent the full Unicode character set.

The default encoding for XML is UTF-8 which is a way of transforming all Unicode characters into a variable length encoding of bytes. It has the advantages that the Unicode characters corresponding to the familiar 7-bit ASCII set have the same byte values as ASCII, and that Unicode characters transformed into UTF-8 can be used with much existing software without extensive software rewrites.

If any other encoding is used then it must be specified in the initial XML declaration. 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">
```

The use of the correct encoding attribute is very important, as the validity of the Metadata File relies on it only containing correctly encoded Unicode characters.

Formatting

Recommendations

An XML file does not need to have any formatting, but in order to improve the readability of the Metadata 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.

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 Metadata File.

a. `<!--`*`comment_text`*`-->` can be used anywhere (after the initial `<?xml` …> declaration) to indicate parts of the Metadata 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 2011-->`

Hierarchical Data

The optional "hierarchy" structure is intended to enable Triple-S to represent "hierarchical" data sets. A hierarchical data set is one in which some questions are asked of, and represent, a "shared" level of information, and other questions deal with the "lower level" individual cases. The classic example of this is a survey where some questions represent household information (location, type of housing, total household income, etc.) and others are asked of each member of the household (age, sex, occupation, personal income, etc.). Some of the analysis of the study may be purely on the basis of household information (and the base is a number of households), some may be of personal information (with a base of number of people), and some may be mixed. There might, for instance, be tables of:

Type of Housing by Region

Type of Housing by Age of Individual

Employment Status by Age of Individual

Household Size (derived from counting people) by Region

Household Size by Age of Individual

Some of these are tables which represent one "level" of the hierarchy, but using data that comes from questions that are asked at another "level".

Packages differ greatly in how they handle this. Many MR packages internally model the notion of hierarchy, and can deal with all the information at the same time. Other MR packages, and most statistical packages, do not have this concept – their input is considered a "flat file", and all data items represent the same entity. To analyse a hierarchical structure in these it is necessary to "flatten" the hierarchy, providing files for analysis at each level (and possibly with information transferred from one level to another).

Triple-S version 2.0 can support hierarchical data structures. It does this by assuming that the exporter will write a separate Triple-S Metadata File and Data File for each level of the hierarchy. These files – with the proper number of cases for each level – can be used directly for analysis by "non-hierarchical" packages. The exporter can (but is not required to) assist this by duplicating data and variables from one level of the hierarchy to another. Packages that can support hierarchical data can reassemble the hierarchy as the Triple-S `<hierarchy>` element will allow them to recreate the levels and dependencies. In particular, the `<hierarchy>` structure identifies “linking” variables, so that every entity at a “child” level points to its parent at a “higher” level. (In our example, data for individuals would include household number (or ID), as a variable that appears in both the household and the individual Triple-S data sets).

As each level of the hierarchy has its own Triple-S Metadata File and Data File, they can be distributed independently. The full hierarchical data set is simply the collection of these, together with a separate file that contains the `<hierarchy>` element and its substructure. This short file simply links together the existing files. It has no data of its own.

Triple-S Names

A Triple-S Metadata File contains a number of “names”, such as the names of the variables or the hierarchy level identifiers. In order to be generally useful these “names” have a restricted definition:-

- names must start with a letter (A-Z or a-z) or _ (underscore) character.
- subsequent characters can be letters (A-Z or a-z), digits (0-9), _ (underscore), or . (period) characters.
- names are case sensitive (i.e. upper and lower case are different)
- names must be unique within their type[7]
- names must not begin, or end, with whitespace (e.g. spaces, tabs), neither should they contain embedded whitespace
- names may be unlimited in length

Although the definition of Triple-S names is described above, most systems that import Triple-S files will also have their own limits and restrictions. The most common will be a maximum size, not case sensitive, and a more restricted set of characters used for e.g. variable names. In these cases the importer may have to generate new names that conform to their own limits and restrictions.

[7] Note that the uniqueness restriction only applies to each type of name. So that e.g. variable names must be unique within a survey or hierarchy, but may be the same as the names used for the hierarchy level identifiers.

Triple-S Numbers

A Triple-S Metadata File will contain many instances of attribute values that are “numbers”. These are either integers, or real numbers which are written with a decimal point.

In order to be generally useful, all integer numbers that are used as attribute values are restricted to signed 32-bit integer values (i.e. -2147483648 to 2147483647).

This limit affects at least the `ident` attribute in a `<variable>` element, the locations within a `<position>` element, integral values within a `<range>` element, and a numeric `code` and `score` within a `<value>` element.

Note that many integer numbers (e.g. the `ident` attribute, the locations within a `<position>` element) are further restricted to positive integer values (i.e. 1 to 214783647).

Real numbers must have a decimal point, plus zero or more decimal places. They are not limited in value. They may be used as the value of the `score` attribute in a `<value>` element, and as the attribute values within a `<range>` and `<value>` elements for a variable of type `quantity.`

General Metadata File Elements and Attributes

This section describes the syntax and function of each of the elements and attributes used to describe the Triple-S XML Metadata File itself. The elements and attributes are shown in the order they are expected in the file.

```
<sss          version="sss_version"
             [ languages="language_list" ]
             [ modes="mode_list" ] >
```

The `<sss>` element is always required and is used to encapsulate the entire specification document. It contains a mandatory attribute `version` and optional `languages` and `modes` 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** or **1.2** then only elements and attributes from the Triple-S XML version 1.1 or 1.2 standard are used. If the *sss_version* is **2.0** then the definition complies with the version 2.0 standard, and new elements and attributes from the Triple-S XML version 2.0 standard may be present.

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 **Specialised Texts** for more on surveys that have multi-lingual texts, for example:

```
<sss version="2.0" languages="en fr">
```

The `modes` attribute is used to indicate that there are specific texts within the Triple-S definition for interviewing and analysis. See the later section on **Specialised Texts** for more on surveys that have modal texts, for example:

```
<sss version="2.0" modes="analysis">
```

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

Optional. The *`date_text`* should represent the date the file was created, for example:

```
<date>20 July 2011</date>
```

Note: there is no prescribed formatting for the *`date_text`*.

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

Optional. The *`time_text`* should represent the time the file was created, for example:

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

Note: there is no prescribed formatting for the *`time_text`*.

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

Optional. The *`origin_text`* should describe the originating system (program and operating system), for example:

```
<origin>MyProg v3, Windows 7</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>
```

Following the above elements, either a `<survey>` or a `<hierarchy>` element describing the actual content of the Metadata File should appear. A `<survey>` element (see the section **Survey Elements and Attributes**) describes either a simple flat set of data, or a Data File that forms part of a hierarchy. A `<hierarchy>` element (see the section **Hierarchy Elements and Attributes**) describes the overall structure of a hierarchical set of Data

Files. Note that a Triple-S Metadata File cannot contain both `<survey>` and `<hierarchy>` elements.

`</sss>`

Mandatory. Finishes the Metadata File.

Survey Elements and Attributes

This section describes the syntax and function of each of the elements and attributes used to form the content of a Triple-S XML `<survey>` element. The elements and attributes are shown in the order they are expected in the file.

`<survey>`

Mandatory. Introduces details of the data for a flat survey, or of a Data File that forms part of a hierarchical survey.

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

Optional. See the section on **Triple-S Names** for the definition of the *`survey_name`*. For those systems with no specific survey naming convention this element could be used to hold the filename, for example:

```
<name>SP1025</name>
```

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

Optional. The version number of the survey, 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/>` elements to indicate new line breaks. For example:

```
<title>Fitness Survey<br/>First wave</title>
```

The title may also contain language-specific and mode-specific texts. These are described in a later section on **Specialised Texts**.

```
<record       ident="record_ident"
            [ href="datafile_uri" ]
            [ format="record_fmt" ]
            [ skip="n" ] >
```

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_ident`* is any single character A to Z or a to z.

The optional *datafile_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 Metadata File and Data File are moved.

For example: **`<record ident="A" href="responses.asc">`**

The optional *record_fmt* can be used to declare the format of the Data File that corresponds to this specification. The default format is fixed format fields, but if specified then it must be one of:

csv	the data representation is comma separated values, using one field for each variable, with data values similar to the fixed format. Note that for data in `csv` format the `position` element refers to the field number.
fixed	the data representation is fixed format fields (the only format supported by previous versions of the Triple-S standard).

The optional `skip` attribute can be used to ignore one or more initial records in the Data File. This will be most useful for csv Data Files where the first line is often used as documentation (e.g. names for the columns/fields) for the succeeding values.

For example: **`<record ident="A" format="csv" skip="1">`**

The *`record_ident`* can be used in conjunction with the *`variable_ident`* (see the `<variable>` element later) to generate unique variable names on import.

Following the `<record>` element, for each variable being described there must be a `<variable>` element:

```
<variable     ident="variable_ident"
              type="variable_type"
            [ use="use_type" ]
            [ format="variable_fmt" ]
```

Mandatory. The *`variable_ident`* is a positive number with or without leading zeroes. Each *`variable_ident`* must be unique within the containing `<record>` element.

The *`variable_type`* must be one of:

single	categorical with one response allowed
multiple	categorical with any number of responses
quantity	numeric value (integer number or real, decimal number)
character	character value
logical	Yes/No or True/False value
date	variable contains a date. The date value must be stored in the YYYYMMDD basic ISO 8601 format.
time	variable contains a time. The time value must be stored in the HHMMSS basic ISO 8601 format.

The *use_type* is optional and describes the role of this variable in the survey. Only a subset of variable types may have a use attribute (as detailed below), and the *use_type* 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 a positive integer `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.

The *variable_fmt* is optional and can be used to declare the format of the codes for this variable. Only variables of type `single` may have a `format` attribute. The default format for all variables of type `single` is **numeric**, but if specified for a variable then it must be one of:

literal	all the codes for this variable are to be treated as characters, rather than numbers. Literal codes are case-sensitive (i.e. "a" and "A" are different).
numeric	all the codes for this variable are to be treated as numbers.

For example:

`<variable ident="10" type="single">`

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

or: **`<variable ident="7" type="single" format="literal">`**

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

Mandatory. The *`name_text`* should represent the name of the variable in the survey. See the section on **Triple-S Names** for the definition of the *`name_text.`*

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

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

Mandatory. The *`label_text`* should represent the label or question text for the original variable, for example:

`<label>First visited</label>`

The text may optionally include any number of `<br/>` elements to indicate new line breaks. The label element may also contain language-specific and mode-specific texts (see later section on **Specialised Texts**).

```
<position     start="start_location"
            [ finish="finish_location" ] />
```

Mandatory. Describes the location of the data values within the data record. The interpretation of this element depends on the format of the Data File (see the `record` element earlier):

For fixed format fields:

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 a length that is at least as long as that implied by the `<size>`, `<values>` and `<spread>` elements.

The parts of the data record defined by the `<position>` elements of different variables may appear in any order, may overlap each other, and do not have to describe the entire data record.

For comma separated values:

The *`start_location`* is a positive integer, which represents the field number, with the first field in the data record being 1.

For example: `<position start="5"/>`

Since the position for a csv file refers to fields, and there is exactly one field per variable, the *`finish_location`* will always be the same as the *`start_location.`* It would therefore be usual for the *`finish_location`* to be omitted where a csv Data File is used. However, importers should not assume that this will be the case as some exports may always explicitly include it.

The fields of the data record defined by the `<position>` elements may appear in any order, may be used more than once, and do not have to describe the entire data record

`<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 one `<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
date	Optional **values** element
time	Optional **values** element

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

Optional element 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. For **fixed** Data Files, 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"/>
```

The `width` attribute may be omitted when used in conjunction with **fixed** data if the *`num_subfields`* exactly fills the space 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`*. Note that for **csv** data the `width` attribute must always be specified as it cannot be determined from the overall width defined in the `<position>` element.

`<values>`

Mandatory for **single**, **multiple** and **quantity** variable types, optional for **date** and **time** variable 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 most one `<range>` element and/or one or more `<value>` elements. If a `<range>` is present then it must be the first element.

```
      <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 element may be followed by any number of `<value>` elements each defining a particular value.

The `<range>` element may not be used when the attribute `format="literal"` is specified on the associated `<variable>` element.

```
<value        code="code_value"
            [ score="score_value" ] >value_text</value>
```

Any number of optional elements that may be used to give labels to specific values of the variable. The *value_text* may optionally include any number of `<br/>` elements to indicate new line breaks. The *value_text* may also contain language-specific and mode-specific texts (see later section on **Specialised 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.

The optional `score` attribute can only be used when the variable is of type **single.** It allows score values to be assigned to the individual code values to be used for computing statistics such as Mean, Standard Deviation etc. The *score_value* must be a number, and may be positive, negative or zero, with or without a decimal point and decimal places. The omission of a `score` implies that records having that value code should be omitted from the base for any statistical computation for that variable.

For single variables:

The *start_value*, *finish_value* and *code_value* for a variable of type **single** depend on whether the attribute `format="literal"` is specified on the `<variable>` element. If this attribute is not present or `format="numeric"` is used or implied, then these codes must all be positive integers or the value zero. However when `format="literal"` is specified, then all *code_values* (even those that look like numbers) are treated as case-sensitive characters, and the `<range>` element cannot be used.

The `<value>` elements do not need to be in any order, nor need they form a complete set with every possible value code present. There is no upper limit to the number of `<value>` elements which may be specified within a 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:

```
   <variable … format="literal">
   . . .
   <values>
   <!—character category codes-->
      <value code="00">Never</value>
      <value code="01">Once a week</value>
      <value code="02">Once a month</value>
      <value code="03">Less frequently</value>
      <value code="X">Don’t know</value>
      <value code="XX">Refused</value>
   </values>
```

Or:

```
   <values>
   <!--with scores-->
      <value code="1" score="2">Very satisfied</value>
      <value code="2" score="1">Satisfied</value>
      <value code="3" score="0">Neither</value>
      <value code="4" score="-1">Unsatisfied</value>
      <value code="5" score="-2">Very unsatisfied</value>
      <value code="9">DK/NS</value>
   </values>
```

For 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 need they form a complete set with every possible value code present. 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">USB Memory Stick</value>
    <value code="2">CD</value>
    <value code="3">DVD</value>
</values>
```

Or:

```
<values>
<!--unlabelled with two explicit categories-->
    <range from="1" to="19" />
    <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 type (i.e. integer or real), 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>
```

For a **quantity** variable of type real, the number of decimal places must be the same for all these attribute values within a values element. The number of decimal places must be identical to the number of decimal places used to represent the data in the corresponding Data File.

The *start_value, finish_value, code_value* and *score_value* must contain at least one digit. The table below gives examples of correct and incorrect representations:

Value	
`1`	Correct (integer)
`1.0`	Correct (real)
`+1.0`	Incorrect - 'plus' sign not allowed
`-1.0`	Correct (real)
`-  1.0`	Incorrect - contains embedded spaces
`1.`	Correct (real)
`.1`	Correct (real)
`-.1`	Correct (real)
`-.`	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.

For date and time variables:

The *`start_value`*, *`finish_value`* and *`code_value`* explicitly define the valid range. Note that the format for **date** and **time** variables is fixed (YYYYMMDD for dates and HHMMSS for times). The valid range for a variable of type **date** or **time** must conform to this format.

For example:

```
<values>
<!--dates within 2011-->
    <range from="20110101" to="20111231" />
</values>
```

`</values>`

Mandatory if there is a `<values>` element. Completes the description of the valid values for the variable.

For character variables:

`<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:

```
<!--100 characters maximum-->
<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.

Hierarchy Elements and Attributes

This section describes the syntax and function of each of the elements and attributes used to form the content of a Triple-S XML `<hierarchy>` Metadata File. The elements are shown in the order they are expected in the file.

`<hierarchy>`

Mandatory. Introduces details of the hierarchical structure.

```
<level       ident="level_ident"
             href="metadata_uri" >
```

Mandatory. One level element appears for each level in the hierarchy. It is used to associate a *`level_ident`* with the *`metadata_uri`* of the Triple-S Metadata File describing the variables (and linking to the associated data) for the level. See the section on **Triple-S Names** for the definition of the *`level_ident`*.

For example:

```
<level ident="hhold" href="hhold.sss">
```

Any level specification can optionally include one or more parent specifications as described below. Note that one or more levels can be 'root' levels with no enclosed parent specification.

```
<parent      parlev="parent_level_ident"
             linkvar="link_variable_name"
           [ ordered="ordered_state" ] >
```

Optional. One or more parent elements may be defined for each level element.

The *`parent_level_ident`* is the *`level_ident`* of the associated parent level. For example, in a household-person hierarchy, the *`parent_level_ident`* of the person level would be the *`level_ident`* of the household level.

The *`link_variable_name`* is the name of the linking variable that appears in both child and parent data sets. It must be a **quantity** or **character** type variable. This variable must have the same name in the different (parent and child) data sets. The variable used as the linking variable must, at the “parent” level, have unique and unduplicated data values in each record (that is, no two records can have the same value).

The *`ordered_state`* is optional and identifies whether the data in the corresponding level is ordered or not. It must be one of:

yes	data in the parent level is ordered
no	data in the parent level should be treated as unordered

If the *ordered_state* is not explicitly specified the value "no" is assumed.

In particular, when dealing with hierarchical data, please note:

1. The specification **ordered="yes"** does not imply an alphanumerical or other absolute ordering, just that the records at this level are arranged in the same order as those in the parent level.

2. Aside from downwardly propagated variables, all variables in the combined set of definitions must have unique names. This has the implication that every individual file must itself have unduplicated names.

3. It is theoretically possible to have a circular list of parents, and implementers should guard against the potential of infinite recursion.

`</hierarchy>`

Mandatory. Finishes the definition for the hierarchy structure.

Specialised Texts

The text for a `survey <title>`, `variable <label>` or `values <value>` element may contain any number of specialised text elements. These are in addition to the plain or default text that may or may not be present for these elements. There are two types of specialised texts:

Languages

The use of multiple language texts within a specification must be signalled by a list of the language identifiers that are used. This is done by adding a `languages="`*`language_identifier_list`*`"` attribute on the initial `<sss>` element.

For example:

```
<sss version="2.0" languages="en-GB en-US fr">
```

`<text xml:lang="``language_ident`**`">`***`formatted_text`***`</text>`**

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

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>
```

Although there is no restriction on `language_ident`, 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 3066, *Tags for the Identification of*

Languages (http://www.ietf.org/rfc/rfc3066.txt) or its successor on the Standards Track.

Note:

IETF RFC 3066 tags are constructed from two and three-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*."

Modes

The use of specialised texts for interviewing and/or analysis within the specification must be signalled by a `modes="`*`mode_identifier_list`*`"` attribute on the initial `<sss>` element.

Two explicit modes are available: “interview” and “analysis”. In the absence of a mode specification, the appropriate text is assumed to be used in both modes.

For example:

```
<sss version="2.0" modes="analysis">
```

<text mode="mode_identifier**">***formatted_text***</text>**

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

For example:

```
<label>How old are you?
    <text mode="analysis">Age of respondent</text>
</label>
```

The standard does not support the use of embedded HTML within texts (with the one exception of `<br/>` elements to indicate new line breaks). Exporters should remove any HTML when generating texts, in particular for the “analysis” mode.

The `language` and `mode` attributes may be combined if the appropriate `languages` and `modes` attributes appear on the `<sss>` element.

For example:

```
<label>Age
    <text xml:lang="en-GB" mode="interview">How old are you?</text>
    <text xml:lang="fr" mode="interview">Quel est votre âge?</text>
    <text xml:lang="en-GB" mode="analysis">Age of respondent</text>
    <text xml:lang="fr" mode="analysis">Âge de répondant</text>
</label>
```

The Data File

Overview

The Data File is composed of individual records. Each record contains the responses for each of the variables in the corresponding Metadata File given by one respondent. Note there is no requirement that all data in the Data File records be described by the associated Metadata File.

The individual records can consist of either fixed format fields or comma separated values. All records in the Data File must be of the same type (i.e. all `fixed` or all `csv`).

Data is recorded in fields and arranged in the manner defined by the `<position>` elements of the variables in the Metadata File. The type and other definitions for the corresponding variable determine the interpretation of each field.

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.

Basic Formatting Rules

1. Other than the record terminator (see below), only characters in the range decimal 32 to 255 are valid (i.e. the Data File is always ISO-8859-1 regardless of the encoding used in the Metadata File).

2. 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.

3. 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.

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

5. There is no maximum record length.

Fixed Format Files

The Data File is composed of fixed format records. If any record is shorter than the highest location defined in a `<position>` element then the extra columns should be treated as blank.

Data is recorded in fields of fixed length and arranged in the manner defined by the `start` and `finish` attributes of the `<position>` elements for the variables in the Metadata File. The type and other definitions for the corresponding variable determine the interpretation of each field.

CSV Files

The Data File is composed of records of varying length. All records should contain the same number of values which must be at least as many as the highest field defined in a `<position>` element.

Data is recorded in fields that generally follow the style generated by the Excel spreadsheet program. The following summarises the format of a Triple-S `csv` Data File:

1. Each record is one line and may not contain embedded line-breaks (even within a quoted string in a field).

2. Data fields are separated with commas.

3. Leading and trailing space-characters adjacent to comma field separators are ignored.

4. Character data fields with embedded commas must be delimited with double-quote characters.

5. Character data fields that contain double quote characters must be surrounded by double-quotes, and the embedded double-quotes must each be represented by a pair of consecutive double quotes.

6. Data fields with leading or trailing spaces must be delimited with double-quote characters.

7. A data field representing a bit-style multiple which begins with "0" (zero) should always be delimited with double-quote characters.

8. Any data field may be delimited with double quotes. The delimiters will always be discarded.

9. The initial records in a `csv` file may be header records containing items such as column (field) names.

Individual Data Items

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

Note: In the following tables the character “b” is used in the data record column to represent a space (blank column), and the character “x” indicates a data record column that should contain either a space or zero character.

Variables of type Single

Data for Singles may be recorded as either numeric codes or literal strings.

Numeric codes

Data is recorded as an integer number or 0 (zero) as described by the `<values>` element.

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.

For example:

Data value	Maximum in `<values>` element	`<position>` element	Data record b=space
7	9	start="21" finish="21"	7
7	9	start="21" finish="22"	07 or b7
7	99	start="21" finish="22"	07 or b7
7	99	start="21"	illegal
7	99	start="21" finish="24"	0007 or bbb7
17	99	start="21" finish="22"	17
17	99	start="21" finish="24"	0017 or bb17
142	9999	start="21" finish="24"	0142 or b142
missing	9999	start="21" finish="24"	bbbb
7	99	start="4" (csv format)	7 or 07 or b7

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. Export programs must ensure that any extra columns contain leading blanks or zeros.

Literal strings

Data is recorded as characters (even if the code is numeric) as described by the `<values>` element. The literal codes are case-sensitive and may contain blanks.

The data field length is derived from the `<value>` elements contained within the `<values>` element, and is the minimum number of characters required to represent the longest literal. If a particular data value requires less than the maximum for the field, it should be left justified using space characters as padding.

For example:

Data value	Maximum code length in `<values>` element	`<position>` element	Data record b=space
`A`	`1`	`start="21" finish="21"`	`A`
`A`	`1`	`start="21" finish="22"`	`Ab`
`A`	`2`	`start="21" finish="22"`	`Ab`
`A`	`2`	`start="21"`	`illegal`
`A`	`2`	`start="21" finish="24"`	`Abbb`
`ZZ`	`2`	`start="21" finish="22"`	`ZZ`
`ZZ`	`2`	`start="21" finish="24"`	`ZZbb`
`missing`	`4`	`start="21" finish="24"`	`bbbb`
`A`	`1`	`start="4" (csv format)`	`A or "A"`
`A`	`2`	`start="4" (csv format)`	`A or "A" or "A "`

If the data field length from the `<value>` element is less than that defined in the corresponding `<position>` element then it is assumed to be left justified within the locations defined in the `<position>` element. Export programs must ensure that any extra columns contain blanks.

Note that in a CSV Data File any literal value with embedded commas, or leading/trailing spaces, must be delimited with double-quote characters

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. Export programs should ensure that any extra columns contain blanks or zeros.

Note that in a CSV Data File any data field representing a bit-style multiple which begins with "0" (zero) should always be delimited with double-quote characters.

For example:

Data value	Maximum in `<values>` element	`<position>` element	Data record b=space, x=space or zero
`1`	`1 to 9`	`start="21" finish="29"`	`100000000`
`1`	`1,2,3 and 9`	`start="21" finish="29"`	`100xxxxx0`
`1, 3`	`1 to 12`	`start="21" finish="32"`	`101000000000`
`none`	`1 to 99`	`start="21" finish="120"`	`000000000...0`
`2, 8`	`1 to 9`	`start="21" finish="30"`	`010000010b or 0100000100`
`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 10`	`start="21" finish="30"`	`bbbxxxxxxb`
`1`	`1 to 9`	`start="5" (csv format)`	`100000000 or "100000000"`
`2, 8`	`1 to 9`	`start="5" (csv format)`	`"010000010"`

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 associated `<values>` or `<range>` elements. 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 value 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. Export programs must ensure that extra columns contain blanks or zeros within 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. Export programs must ensure that any extra columns contain blanks or zeros.

For example:

Data value	Maximum in `<values>` element	`<spread>` element	`<position>` element	Data record b=space
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"	b1b0 or b0b1 or 0100 etc.
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"	20bb or 02bb or 2000 etc.
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
1	1 to 9	subfields="2" width="1"	start="4" (csv format)	1 or 10 or "10" or "01"
1	1 to 99	subfields="2" width="2"	start="4" (csv format)	"0100" or "0001"

Variables of type Quantity

Data is recorded as a number with the same number of decimal places as were used in the `<values>` and `<range>` elements in the specification of the corresponding variable. A decimal point (i.e. full stop or period, '.') should always appear if one was used within the `<values>` element specification.

For example:

Data value	`<range>` element	`<position>` element	Data record b=space
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"	bb7 or b07 or 007
-1.00	from="-1.00" to="99.99"	start="21" finish="26"	b-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
7	from="0" to="99"	start="4" (csv format)	7 or 07 or "07"
-1.00	from="-1.00" to="99.99"	start="4" (csv format)	-1.00 or "-1.00"

The data field length must accommodate the longest allowable value defined by the `<values>` and `<range>` elements. 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. Export programs must ensure that any extra columns contain blanks or zeros.

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>` element. 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`

Variables of type Date

Data is recorded in the YYYYMMDD basic ISO 8601 format where “YYYY” is the 4 digit year, “MM” is the 2 digit month, and “DD” is the 2 digit day.

The length of the field is always 8 characters. If the `<position>` element defines a width of more than 8 characters then the leftmost characters are used and all others should be ignored.

For example, a value of 1st April 2011 would be represented as: `20110401`

Variables of type Time

Data is recorded in the HHMMSS basic ISO 8601 format where “HH” is the 2 digit hour using the 24-hour clock, “MM” is the 2 digit minute, and “SS” is the 2 digit second.

The length of the field is always 6 characters. If the `<position>` element defines a width of more than 6 characters then the leftmost characters are used and all others should be ignored.

For example, a value of 4.15pm would be represented as: `161500`

Examples

Triple-S version 2.0 Metadata File Example 1 (Fixed format raw data)

The example defines a survey with twelve variables each demonstrating one or more v2.0 features as annotated…

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v2.0//EN"
                     "http://www.triple-s.org/dtd/sss_v20.dtd">

<sss version="2.0" modes="interview analysis">
<!-- introducing modes used to specialise texts at Q2 -->

<date>14 April 2011</date>
<time>16:00</time>
<origin>SurveyProg v1.3.05</origin>
<user>User Site</user>

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

    <record ident="V">

        <variable ident="1" type="quantity" use="serial">
        <!-- serial variable with long name -->
            <name>RESPONDENT_ID</name>
            <label>Respondent ID</label>
    <position start="1" finish="6"/>
    <values>
        <range from="000001" to="999999" />
    </values>
</variable>

<variable ident="2" type="date">
<!-- date variable -->
    <name>Q1.a</name>
    <label>Date of visit</label>
    <position start="7" finish="14"/>
    <values>
        <range from="20110101" to="20111231" />
    </values>
</variable>

<variable ident="3" type="time">
<!-- time variable -->
    <name>Q1.b</name>
    <label>Time of visit</label>
    <position start="15" finish="20"/>
    <!-- implicit range from="000000" to="235959" -->
</variable>
```

```
        <variable ident="4" type="single">
            <name>Q2</name>
            <label>Frequency of visit
                <text mode="interview">Have you visited here before?</text>
                <text mode="analysis">Visited before</text>
            </label>
            <!-- specialised texts -->
            <position start="21" />
            <values>
                <value code="0">No, this is the first visit</value>
                <value code="1">I have visited before within the
year</value>
                <value code="2">I visited before that</value>
            </values>
            <!-- value list including code 0 -->
        </variable>

        <variable ident="5" type="multiple">
            <name>Q3</name>
            <label>Attractions visited</label>
            <position start="22" finish="30" />
            <values>
                <value code="1">Sherwood Forest</value>
                <value code="2">Nottingham Castle</value>
                <value code="3">&quot;Friar Tuck&quot; Restaurant</value>
                <!-- quotation marks within label fields -->
                <value code="4">&quot;Maid Marion&quot; Cafe</value>
                <value code="5">Mining museum</value>
                <value code="9">Other</value>
            </values>
        </variable>

        <variable ident="6" type="character">
        <name>Q3.a</name>
        <label>Other attractions visited</label>
        <position start="31" finish="60" />
        <size>30</size>
    </variable>

    <variable ident="7" type="single">
        <name>Q4</name>
        <label>Overall impression</label>
        <position start="61" />
        <values>
            <!-- scores -->
            <value code="1" score="2">Very Good</value>
            <value code="2" score="1">Good</value>
            <value code="3" score="0">OK</value>
            <value code="4" score="-1">Poor</value>
            <value code="5" score="-2">Very poor</value>
            <value code="9">DK/NS</value>
            <!-- missing score -->
        </values>
    </variable>

    <variable ident="8" type="multiple">
        <name>Q5</name>
        <label>Two favourite attractions visited</label>
        <position start="62" finish="63" />
        <spread subfields="2" />
        <!--same answer list as Q3-->
        <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="9" type="quantity">
        <name>Q6</name>
        <label>Miles travelled</label>
        <position start="64" finish="66" />
        <values>
            <range from="1" to="499" />
            <value code="500">500 or more</value>
            <value code="999">Not stated</value>
        </values>
        <!-- labelled values on a quantity -->
    </variable>

    <variable ident="10" type="logical">
        <name>Q7</name>
        <label>Would come again</label>
        <position start="67" />
    </variable>

    <variable ident="11" type="single" format="literal">
        <name>Q8</name>
        <label>When is that most likely to be</label>
        <position start="68" />
        <filter>Q7</filter>
        <!-- filter -->
        <values>
            <value code="A">Within 3 months</value>
            <value code="B">Between 3 months and 1 year</value>
            <value code="C">More than 1 years time</value>
            </values>
            <!-- literal code values -->
        </variable>

        <variable ident="999999" type="quantity" use="weight">
        <!-- weight variable -->
        <!-- large magnitude ident -->
            <name>WT</name>
            <label>Case weight</label>
            <position start="69" finish="75"/>
            <values>
                <range from="0.0000" to="99.9999"/>
            </values>
        </variable>

    </record>

</survey>

</sss>
```

Triple-S version 2.0 Data File Example 1 (Fixed format raw data)

The example Data File is intended to be used in conjunction with the Metadata File above.

```
520001201105041120000101010001Nottingham Goose Fair          251 251A 1.1310
520002201105061343002010000000                                92 1000  0.9921
520003201105031805001110000001"Heritage" Zone                 1929991C 1.0089
```

Interpretation

The Triple-S Metadata File applied to the above Data File should result in the following interpretations.

	Respondent 1	Respondent 2	Respondent 3
RESPONDENT_ID	520001	520002	520003
Date of visit	May 4th, 2011	May 6th, 2011	May 3rd, 2011
Time of visit	11:20 am	1:43 pm	6:05 pm
Visited before	No, first visit	Visited before that	Visited before within the year
Attractions visited	Sherwood Forest, "Friar Tuck" Restaurant, Mining Museum, Other	Nottingham Castle	Sherwood Forest, Nottingham Castle, Other
Other attractions	Nottingham Goose Fair	(no reply)	"Heritage" Zone

Overall impression	Good	DK/NS	Very Good
Two favourite attractions	Mining Museum, Sherwood Forest	Nottingham Castle	Other, Nottingham Castle
Miles travelled	25	100	Not stated
Would come again	true	false	true
When come again	Within 3 months	(not asked)	More than 1 year
Case weight	1.131	0.9921	1.0089

Triple-S version 2.0 Metadata File Example 2 (CSV raw data)

The example defines a survey with twelve variables each demonstrating one or more v2.0 features as annotated…

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v2.0//EN"
"http://www.triple-s.org/dtd/sss_v20.dtd">

<sss version="2.0" modes="interview analysis">
<!-- introducing modes used to specialise texts at Q2 -->

<date>14 April 2011</date>
<time>16:00</time>
<origin>SurveyProg v1.3.05</origin>
<user>User Site</user>

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

    <record ident="V" format="csv" skip="1">
    <!-- csv file specification: skip first record, default name .csv -->

        <variable ident="1" type="quantity" use="serial">
        <!-- serial variable with long name -->
            <name>RESPONDENT_ID</name>
            <label>Respondent ID</label>
            <position start="1"/>
            <values>
                <range from="000001" to="999999" />
            </values>
        </variable>

        <variable ident="2" type="date">
        <!-- date variable -->
            <name>Q1.a</name>
            <label>Date of visit</label>
        <position start="2"/>
        <values>
            <range from="20110101" to="20111231" />
        </values>
    </variable>

    <variable ident="3" type="time">
    <!-- time variable -->
        <name>Q1.b</name>
        <label>Time of visit</label>
        <position start="3"/>
        <!-- implicit range from="000000" to="235959" -->
    </variable>

    <variable ident="4" type="single">
        <name>Q2</name>
        <label>Frequency of visit
            <text mode="interview">Have you visited here before?</text>
            <text mode="analysis">Visited before</text>
        </label>
        <!-- specialised texts -->
        <position start="4" />
        <values>
            <value code="0">No, this is the first visit</value>
            <value code="1">I visited before within the year</value>
            <value code="2">I visited before that</value>
        </values>
        <!-- value list including code 0 -->
    </variable>

    <variable ident="5" type="multiple">
        <name>Q3</name>
        <label>Attractions visited</label>
        <position start="5" />
        <values>
            <value code="1">Sherwood Forest</value>
            <value code="2">Nottingham Castle</value>
            <value code="3">&quot;Friar Tuck&quot; Restaurant</value>
            <!-- quotation marks within label fields -->
            <value code="4">&quot;Maid Marion&quot; Cafe</value>
            <value code="5">Mining museum</value>
            <value code="9">Other</value>
        </values>
    </variable>
```

```
<variable ident="6" type="character">
    <name>Q3.a</name>
    <label>Other attractions visited</label>
    <position start="7" />
    <!-- positions do not need to be consecutive -->
    <size>30</size>
</variable>

<variable ident="7" type="single">
    <name>Q4</name>
    <label>Overall impression</label>
    <position start="6" />
    <values>
        <!-- scores -->
        <value code="1" score="2">Very Good</value>
        <value code="2" score="1">Good</value>
        <value code="3" score="0">OK</value>
        <value code="4" score="-1">Poor</value>
        <value code="5" score="-2">Very poor</value>
        <value code="9">DK/NS</value>
        <!-- missing score -->
    </values>
</variable>

<variable ident="8" type="multiple">
    <name>Q5</name>
    <label>Two favourite attractions visited</label>
    <position start="8" />
    <spread subfields="2" width="1" />
    <!-- spread fields explicitly define width with CSV data -->
    <!-- same answer list as Q3 -->
    <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="9" type="quantity">
    <name>Q6</name>
```

```
            <label>Miles travelled</label>
            <position start="9" />
            <values>
                <range from="1" to="499" />
                <value code="500">500 or more</value>
                <value code="999">Not stated</value>
            </values>
            <!-- special values on a quantity -->
        </variable>

        <variable ident="10" type="logical">
            <name>Q7</name>
            <label>Would come again</label>
            <position start="10" />
        </variable>

        <variable ident="11" type="single" format="literal">
            <name>Q8</name>
            <label>When is that most likely to be</label>
            <position start="11" />
            <filter>Q7</filter>
            <!-- filter -->
            <values>
                <value code="A">Within 3 months</value>
                <value code="B">Between 3 months and 1 year</value>
                <value code="C">More than 1 years time</value>
            </values>
            <!-- literal code values -->
        </variable>

        <variable ident="999999" type="quantity" use="weight">
        <!-- weight variable -->
        <!-- large magnitude ident -->
            <name>WT</name>
            <label>Case weight</label>
            <position start="12" />
            <values>
                <range from="0.0000" to="99.9999"/>
            </values>
        </variable>

    </record>
</survey>
</sss>
```

Triple-S version 2.0 Data File Example 2 (CSV raw data)

The example Data File is intended to be used in conjunction with the Metadata File above.

```
RESPONDENT_ID,Q1.a,Q1.b,Q2,Q3,Q4,Q3.a,Q5,Q6,Q7,Q8,WT
520001,20110504,112000,0,101010001,2,Nottingham Goose Fair,51,25,1,A,1.131
520002,20110506,134300,2,"010000000",9,,2,100,0,,0.9921
520003,20110503,180500,1,110000001,1,"""Heritage"" Zone",92,999,1,C,1.0089
```

Interpretation

The Triple-S Metadata File applied to the above Data File should result in the following interpretations.

	Respondent 1	**Respondent 2**	**Respondent 3**
RESPONDENT_ID	520001	520002	520003
Date of visit	May 4th, 2011	May 6th, 2011	May 3rd, 2011
Time of visit	11:20 am	1:43 pm	6:05 pm
Visited before	No, first visit	Visited before that	Visited before within the year
Attractions visited	Sherwood Forest, "Friar Tuck" Restaurant, Mining Museum, Other	Nottingham Castle	Sherwood Forest, Nottingham Castle, Other
Other attractions	Nottingham Goose Fair	(no reply)	"Heritage" Zone

Overall impression	Good	DK/NS	Very Good
Two favourite attractions	Mining Museum, Sherwood Forest	Nottingham Castle	Other, Nottingham Castle
Miles travelled	25	100	Not stated
Would come again	true	false	true
When come again	Within 3 months	(not asked)	More than 1 year
Case weight	1.131	0.9921	1.0089

Triple-S version 2.0 Hierarchy File Example

Part 1: the Hierarchical Definition

The example defines a hierarchical definition for three (household, person and trip) surveys.

Each house has zero, one or more people associated with it and each person has zero, one or more trips associated with them. Indirectly, each trip is also associated with a household (the household of the person making the trip).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v2.0//EN"
"http://www.triple-s.org/dtd/sss_v20.dtd">

<sss version="2.0">

<date>14 April 2011</date>
<time>16:30</time>
<origin>SurveyProg v1.3.05</origin>
<user>User Site</user>

<hierarchy>
    <level ident="hhold" href="householddata.sss" />
    <level ident="person" href="persondata.sss">
        <parent level="hhold" linkvar="hnumber" ordered="yes" />
    </level>
    <level ident="trip" href="tripdata.sss">
        <parent level="person" linkvar="pnumber" ordered="yes" />
    </level>
</hierarchy>

</sss>
```

Part 2: The Household Survey

The variables within each of the three surveys are described in three independent Triple-S Metadata Files. Respondent data is held in three associated Data Files. The Metadata Files could be *any* version of Triple-S but are illustrated using version 2.0 syntax.

The "householddata.sss" Metadata File

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

<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v2.0//EN"
"http://www.triple-s.org/dtd/sss_v20.dtd">


<sss version="2.0">


<date>14 April 2011</date>

<time>16:31</time>

<origin>SurveyProg v1.3.05</origin>

<user>User Site</user>


<survey>

    <name>PR9012-HOUSEHOLD</name>

    <title>Regional Travel Survey<br/>Households</title>


    <record ident="V">


        <variable ident="1" type="quantity" use="serial">

        <!-- the linkvar has the attribute use="serial" -->

            <name>hnumber</name>

            <label>Household</label>

            <position start="1" finish="6"/>

            <values>

                <range from="000001" to="999999" />

            </values>
        </variable>

        <variable ident="2" type="single">
            <name>htype</name>
            <label>House type</label>
            <position start="7"/>
            <values>
                <value code="1">Flat or Maisonette</value>
                <value code="2">Terraced House</value>
                <value code="3">Semi-detached House</value>
                <value code="4">Detached House</value>
            </values>
        </variable>

        <variable ident="3" type="single">
            <name>hlocation</name>
            <label>Household location</label>
            <position start="8"/>
            <values>
                <value code="1">North</value>
                <value code="2">South</value>
                <value code="3">East</value>
                <value code="4">West</value>
            </values>
        </variable>

    </record>
</survey>
</sss>
```

The "householddata.asc" raw Data File

The example Data File is intended to be used in conjunction with the Metadata File above.

```
01000123
01000232
01000313
```

Interpretation of the Household survey data

The Triple-S Metadata File applied to the above Data File should result in the following interpretations.

	Household 1	**Household 2**	**Household 3**
hnumber	010001	010002	010003
htype	Terraced House	Semi-detached House	Flat or Maisonette
hlocation	East	South	East

Part 2: The Person Survey

As before, the variables are described in the Triple-S Metadata File and respondent data is in the associated Data File. The Metadata Files could be *any* version of Triple-S but are illustrated using version 2.0 syntax.

The "persondata.sss" Metadata File

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v2.0//EN"
"http://www.triple-s.org/dtd/sss_v20.dtd">
<sss version="2.0">

<date>14 April 2011</date>
<time>16:31</time>
<origin>SurveyProg v1.3.05</origin>
<user>User Site</user>

<survey>
    <name>PR9012-PERSON</name>
    <title>Regional Travel Survey<br/>Persons</title>

    <record ident="V">
        <variable ident="1" type="quantity">
            <name>hnumber</name>
            <label>Household</label>
            <position start="1" finish="6"/>
            <values>
                <range from="000001" to="999999" />
            </values>
        </variable>
        <variable ident="2" type="quantity" use="serial">
            <name>pnumber</name>
            <label>Person</label>
            <position start="1" finish="8"/>
            <values>
                <range from="00000001" to="99999919" />
            </values>
        </variable>
        <variable ident="3" type="single">
            <name>pgender</name>
            <label>Gender</label>
            <position start="9"/>
            <values>
                <value code="1">Male</value>
                <value code="2">Female</value>
            </values>
        </variable>
        <variable ident="4" type="single">
            <name>page</name>
            <label>Age</label>
            <position start="10"/>
            <values>
                <value code="1">Under 21</value>
                <value code="2">21 to 45</value>
                <value code="3">46 to 65</value>
                <value code="4">Over 65</value>
            </values>
        </variable>
    </record>
</survey>
</sss>
```

The "persondata.asc" raw data file

The example Data File is intended to be used in conjunction with the Metadata File above.

```
0100010122
0100010212
0100020114
0100020223
0100020311
0100030122
```

Interpretation of the Person survey data

The Triple-S Metadata File applied to the above Data File should result in the following interpretations.

	Household 1		**Household 2**			**Household 3**
	Person 1	**Person 2**	**Person 1**	**Person 2**	**Person 3**	**Person 1**
hnumber	010001	010001	010002	010002	010002	010003
pnumber	01000101	01000102	01000201	01000202	01000203	01000301
pgender	Female	Male	Male	Female	Male	Female
Page	21 to 45	21 to 45	over 65	46 to 65	Under 21	21 to 45

Part 3: The Trip Survey

As before, the variables are described in the Triple-S Metadata File and respondent data is in the associated Data File. The Metadata Files could be *any* version of Triple-S but are illustrated using version 2.0 syntax.

The "tripdata.sss" Metadata File

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sss PUBLIC "-//triple-s//DTD Survey Interchange v2.0//EN"
"http://www.triple-s.org/dtd/sss_v20.dtd">


<sss version="2.0">


<date>14 April 2011</date>
<time>16:32</time>
<origin>SurveyProg v1.3.05</origin>
<user>User Site</user>


<survey>
    <name>PR9012-TRIP</name>
    <title>Regional Travel Survey<br/>Trips</title>

    <record ident="V">

        <variable ident="1" type="quantity">
            <name>pnumber</name>
            <label>Person</label>
            <position start="1" finish="8"/>
            <values>
                <range from="00000001" to="99999919" />
            </values>
        </variable>
        <variable ident="2" type="single">
            <name>tpurpose</name>
            <label>Trip purpose</label>
            <position start="9"/>
            <values>
                <value code="1">Social, domestic, pleasure</value>
                <value code="2">To or from place of work</value>
                <value code="3">Business</value>
            </values>
        </variable>

        <variable ident="3" type="single">
            <name>tmode</name>
            <label>Modes of travel</label>
            <position start="10" />
            <values>
                <value code="1">Car or van driver</value>
                <value code="2">Car or van passenger</value>
                <value code="3">Bus</value>
                <value code="4">Train</value>
                <value code="5">Cycle</value>
                <value code="6">Other</value>
            </values>
        </variable>

    </record>
</survey>
</sss>
```

The "tripdata.asc" Data File

The example Data File is intended to be used in conjunction with the Metadata File above.

```
0100010113
0100010112
0100010224
0100010232
0100010224
0100010211
0100020121
0100020121
0100020111
0100020312
0100030123
0100030123
```

Interpretation of the Trip survey data

The Triple-S Metadata File applied to the above Data File should result in the following interpretations.

	Household 1					
	Person 1	**Person 1**	**Person 2**	**Person 2**	**Person 2**	**Person 2**
pnumber	01000101	01000101	01000102	01000102	01000102	01000102
tpurpose	Social	Social	Work	Business	Work	Social
tmode	Bus	Car Passenger	Train	Car Passenger	Train	Car Driver

	Household 2				Household 3	
	Person 1	**Person 1**	**Person 1**	**Person 3**	**Person 1**	**Person 1**
pnumber	01000201	01000201	01000201	01000203	01000301	01000301
tpurpose	Work	Work	Social	Social	Work	Work
tmode	Car Driver	Car Driver	Car Driver	Car Passenger	Bus	Bus

Organising Hierarchical Imports

Software that is inherently capable of representing hierarchical data structures should have little problem importing and exporting appropriate, separate surveys as outlined in the example.

Software that does not have an in-built hierarchical mechanism could still offer support for hierarchical surveys by generating a survey representing a "flattened" portion (or all) of the hierarchical group of surveys.

Taking the Household, Person, Trip hierarchy described previously, it is possible that a user wants to do analyses of people and the types of housing they live in. That is they need to import and combine elements from both the Household and Person surveys. Doing that with the examples given would result in the following combined survey:

	Household 1		**Household 2**			**Household 3**
	Person 1	**Person 2**	**Person 1**	**Person 2**	**Person 3**	**Person 1**
hnumber	010001	010001	010002	010002	010002	010003
htype	Terraced House	Terraced House	Semi-detached House	Semi-detached House	Semi-detached House	Flat or Maisonette
hlocation	East	East	South	South	South	East
pnumber	01000101	01000102	01000201	01000202	01000203	01000301
pgender	Female	Male	Male	Female	Male	Female
page	21 to 45	21 to 45	over 65	46 to 65	Under 21	21 to 45

An analyst wanting to produce a crosstabulation of a person’s age by the area in which they live would simply ask for a crosstabulation of `page` by `hlocation`.

Similarly for other analyses involving "person" variables or combinations of "person" and "household" variables. Although possible, analyses involving just "household" variables would actually be counting or summarising persons and should typically be avoided.

In general, if your software does not directly handle hierarchical structures, provide extended import options which ask for either a list of surveys or a start survey and an end survey. The surveys specified should be interconnected by the `linkvar` variable(s) as appropriate. The import process then results in a survey where records are the most frequently occurring records of the given input surveys and data from other records is duplicated as appropriate (and as outlined above).

The Triple-S XML DTD

The Triple-S XML 2.0 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_v20.dtd.

```
<!-- Triple-S.dtd                                                      -->
<!-- Standalone XML version 2.0                                        -->
<!-- March 2006                                                        -->
<!-- ===========================================================       -->
<!-- An XML definition for moving surveys between packages             -->
<!-- on various hardware and software platforms.                       -->
<!-- ===========================================================       -->
<!-- This DTD has been produced by the Triple-S Group                  -->
<!-- (Laurance Gerrard, Keith Hughes, Steve Jenkins, Ed Ross           -->
<!-- and Geoff Wright).                                                -->
<!-- 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 v2.0//EN                    -->
<!-- Public URL:                                                       -->
<!--   http://www.triple-s.org/dtd/sss_v20.dtd                         -->
<!-- ===========================================================       -->
<!--                                                                   -->
<!-- Version history:                                                  -->
<!--                                                                   -->
```

```
<!--   1.0            Original non-XML Triple-S standard            -->
<!--   1.1   Feb 2000  XML version based on non-XML 1.1 standard     -->
<!--   1.2   Jun 2002  XML version 1.2 (initial version)             -->
<!--   2.0   Feb 2005  XML version 2.0 (initial version)             -->
<!--         May 2005  XML version 2.0 (including hierarchy)         -->
<!--         Mar 2006  XML version 2.0 (final version)               -->
<!--                                                                 -->
<!-- ===========================================================     -->
<!--                                                                 -->
<!--          BEGINNING OF ACTUAL DOCUMENT TYPE DEFINITION           -->
<!--                                                                 -->



<!-- parameter entities                                              -->
<!ENTITY % vartype "single |
                    multiple |
                    quantity |
                    character |
                    logical |
                    date |
                    time" >
<!ENTITY % usetype "serial | weight" >
<!ENTITY % recfmt  "fixed | csv" >
<!ENTITY % varfmt  "literal | numeric" >
<!ENTITY % txtmode "interview | analysis" >
<!ENTITY % yesno   "yes | no" >

<!-- formatted text with break element                               -->
<!ELEMENT br EMPTY>
<!ENTITY % formatted_text "#PCDATA | br" >

<!-- multi-lingual texts                                             -->
<!ELEMENT text (%formatted_text;)*>
```

```
<!ATTLIST text xml:lang NMTOKEN #IMPLIED
               mode (%txtmode;) #IMPLIED>
<!ENTITY % texts "(%formatted_text; | text)*">



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



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

    <!ELEMENT time (#PCDATA)>

    <!ELEMENT origin (#PCDATA)>

    <!ELEMENT user (#PCDATA)>



<!--                                                                    -->
<!-- HIERARCHY DESCRIPTION                                              -->
<!-- =====================                                              -->
<!--                                                                    -->
        <!ELEMENT hierarchy (level+)>

        <!ELEMENT level (parent*)>
        <!ATTLIST level ident CDATA #REQUIRED
                        href CDATA #REQUIRED>

        <!ELEMENT parent EMPTY>
        <!ATTLIST parent level CDATA #REQUIRED
                         linkvar CDATA #REQUIRED
                         ordered (%yesno;) #IMPLIED>

<!--                                                                    -->
<!-- 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
                         format (%recfmt;) "fixed"
                         skip CDATA #IMPLIED>

<!--                                                                    -->
<!-- VARIABLE DESCRIPTION                                               -->
<!-- ====================                                               -->
```

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

                <!-- ELEMENT name already defined -->

                <!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
```

```
                                    score CDATA #IMPLIED>


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


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

History of Changes

Changes from Triple-S XML 1.2 to Triple-S XML 2.0

The Triple-S XML version 2.0 standard is based on the XML version 1.2 standard, and subject to the new limits and restrictions should be essentially a superset. The following sections provide a summary of the main additions and changes from XML version 1.2.

1. The standard has been extended to support hierarchical data structures.

2. The Data File may now be in fixed or CSV format.

3. The SKIP attribute has been added to the RECORD element.

4. The valid set of VALUE codes for variables of type SINGLE has been extended to include 0 (zero) and also literal strings of appropriate length.

5. SCORE attributes have been introduced to allow scores to be assigned to values of a variable of type SINGLE.

6. New variable types DATE and TIME have been added.

7. The TEXT element now allows the survey title, variable label and values value elements to include specialised interview and analysis texts.

8. The rules for defining names (SURVEY, VARIABLE and LEVEL) have been restricted, and altered to ensure that all variable names are unique within a RECORD.

9. Integer attribute values are restricted to be 32-bit numbers.

10. The rules for defining data values within the data record have been revised, and examples of CSV format values added.

11. The **options** attribute of the SURVEY element, originally introduced in version 1.1. The only allowable option was **standardnames** and that has been removed because of frequent misuse.

Changes from Triple-S XML 1.1 to Triple-S XML 1.2

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 have been introduced to give more information about the survey itself.

2. The FILTER element has been introduced 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

The Metadata 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

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 Metadata File to be skipped.

2. The **note** statement allows notes to be inserted within the Metadata File.

3. The **standardnames** option assists 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.

ISBN: 0-9546748-3-9

Association for Survey Computing

PO Box 76, Berkeley, Gloucestershire, UK, GL13 9WU