triple-s

survey interchange standard

# Triple-S XML Version 2.0

## Release Notes

May 2006

# Introduction

This is the fifth revision of the Triple-S standard, following versions 1.0 (classic only), 1.1 (classic and XML) and 1.2 (XML only).

Changes to the standard, and new features, have been chosen by the committee following an Open Meeting with users held on 31st March 2004. The committee would like to thank all who attended the meeting, or sent in discussion papers, for their contributions.

Notes on the changes and new features have been ordered in this document as follows:

- Hierarchical Data
- CSV Data Files
- Code Values
- Score Values
- Date/Time Values
- Specialised Texts
- Limits and Restrictions

Examples and the complete Triple-S DTD can be found in the specification document.

This document can be downloaded from www.triple-s.org

The Triple-S committee consists of Ed Ross, Geoff Wright, Keith Hughes, Laurance Gerard and Steve Jenkins.

The Triple-S initiative is supported and funded by the Association for Survey Computing. Details at http://www.asc.org.uk/

May 2006.

## Hierarchical Data

**Extends the Triple-S specification to support hierarchical data structures.**

### *Outline*

In the case of hierarchical data, a Triple-S exporting program may write a control file and a data file for each “level” of the hierarchy, and a separate “control file” specifying how the files are linked. An importing program which supports hierarchical data can read the control file and, thereby, all the individual files and recreate the hierarchy.  An importing program which does not support hierarchical data can read the files for any individual level of the hierarchy, without being aware of the others.  To support use by non-hierarchical programs, exporting programs may reproduce “higher” level variables (both definitions and data) within the lower level Triple-S files, so as to enable non-hierarchical programs to retrieve all data relevant for that level.

### *Example (of a control file)*

`A simple example, where households had people and people had trips:`

```
    <hierarchy>
       <level ident="hhold" href="hhdata.sss" />
       <level ident="person" href="persondat.sss" >
            <parent parlev="hhold" linkvar="hhnumb"  ordered="yes"/>
       </level>
       <level name="trips" filename="tripdat.sss">
              <parent parlev="person" linkvar="persnumb" ordered="yes" />
       </level>
    </hierarchy>
```

### *Features and Benefits*

It requires:

a) that a linking variable appears in both child and parent data sets, which must be a quantity or character variable. These must have the same name in the different data sets, in line with the general requirement ( c) below..

b) that all variables that are “propagated downward” retain the same name, same definition, and same data at the propagated level as at the parent level.

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

d) that any variable used as a 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).

In general, processing might be more efficient if the “child” file is ordered in the same way as the “parent”. If this is the case, ordered=“yes” should be specified. (If not, ordered=“no”, the default, will be assumed). Specifying ordered=“yes” does not imply that the records are sorted in any specific alphabetic or numeric order, merely that the records are in a consistent order as required by the record set.

**One or more levels can be 'root' levels with no parents.**

**It is possible for a level to have more than one parent. In this case the linking variable will generally (but not necessarily) be different for each parent.**

## *Issues and Problems*

This represents an extension of Triple-S, from defining a single file, to defining a linked family of files. This means that the “packaging” of the Triple-S export becomes less trivial, as the package must include all (local) files. This extends the demands caused by the current *href* on the *record* element.

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

## CSV Data Files

**Extends the Triple-S specification to cover a comma separated (CSV) data file format in addition to the current fixed column format. The definition of a valid Triple-S CSV file follows the style generated by the Excel spreadsheet program.**

### *Outline*

There is a new optional *format* attribute on the *<record>* element. The value is either "fixed" or "csv". For compatibility with the previous Triple-S standard, the default format is "fixed".

The data representation is one CSV field (without subfields) for each variable with data values similar to the existing Triple-S standard.

The first line or lines in a CSV data file are sometimes used as documentation for the succeeding values (e.g. names for the columns/fields). An optional *skip* attribute on the *<record>* element can be used to ignore one or more initial lines in the data file.

### *Examples*

The following example is based on a CSV data file with the first record being skipped: -

```
    <record ident="v" format="csv" skip="1">
         <variable ident="1" type="single">
            <name>Q1</name>
            <label>Number of visits</label>
            <position start="1" />
            . . .
        <variable ident="2" type="multiple">
            <name>Q2</name>
            <label>Attractions visited</label>
            <position start="2" />
            . . .
        <!-- ignore field 3 -->
        <variable ident="3" type="character">
            <!-- only ask if Q2 has code 9 -->
            <name>Q3</name>
            <label>Other attractions visited</label>
            <position start="4"/>
            . . .
        <variable ident="4" type="multiple">
            <name>Q4</name>
            <label>Two favourite attractions visited</label>
            <position start="5"/>
            <spread subfields="2" width="1"/>
            . . .
```

With a data file like: -

```
"Visits","Attractions","Dummy","Other attractions","Favourite attractions",. . .
2,101000001,3,"Amusement Park",19, . . .
3,"010000000",1,,20, . . .
2,"1000100001",3,"""Marco's"" Restaurant",94, . . .
```

### *Features and Benefits*

This extension provides a method for specifying CSV data files with data values similar to the existing Triple-S standard  (i.e. one record per case, and one field per value). The format of the CSV data file is based on that generated by the Excel program, which should provide compatibility with most other users of CSV files.

## *CSV file format*

The following summarises the format of a Triple-S CSV data file, and is based on what is generated by the Excel program:

1. Each record is one line and may not contain embedded line-breaks.

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

## *Notes*

The initial support for delimited data values is restrictive – one variable to a field, and fields must be separated by commas.

In the case of a CSV data file the <*position*> element specifies the field number, and with one data value per field the specification only needs a *start* value.

For <*spread*> style multiples the *width* attribute must be specified, as it can not be determined from the overall field width in the data record.

The *skip* attribute can also apply to fixed format data.

Even if any skipped data records contain variable names the import should always use the variable names specified within the Triple-S metadata file

For compatibility with other users of CSV data, an exporter should normally try to keep the number of variables within a CSV data file at 255 or less.

## Code Values

**Extends the valid range of value codes for variables of type single to include 0 (zero) and also any literal string of appropriate length.**

### *Outline*

Hitherto a number of import and export programs have supported an informal extension to the Triple-S standard, whereby value code 0 is allowed on variables of type *single*: this extension is now “legalised”. Additionally, many survey programs allow data to be coded with letters instead of/as well as numbers, and this is now supported by Triple-S.

### *Example*

```
<variable ident="1" type="single" format="literal">
    <name>Q2</name>
    <label>How often do you go to the cinema?</label>
    <position start="3" finish="4" />
    <values>
        <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="A">DK/NS</value>
    </values>
</variable>
```

### *Features and Benefits*

Optionally declaring the format of a codes to be “literal” (at variable level only) allows the exporter greater freedom to describe the data as it is, without having to recode it; allows him to distinguish e.g. between numeric “1” and literal “01” on a two-column field; it warns the importer of a feature he may not be able to handle; and is upwards compatible from previous versions. Note that the default value for this parameter may be explicitly set as format="numeric".

### *Notes*

The extension to allow value code 0 does not require that format=“literal” is used.

When format=“literal” is used, then the codes specified are treated as case-sensitive.

When format=“literal” is used, then the “range” feature may not be used.

When format=“literal” is used, then the value code is to be taken as left-adjusted in the data and blank-filled.

## Score Values

**Allows scores to be assigned to the values of variables of type *single*, to be used (possibly and inter alia) for computing statistics such as Mean, Standard Deviation etc.**

### *Outline*

One numeric score value may be assigned to each value code in a value block. The score may be a number of any magnitude, positive, negative or zero, with or without a decimal point and decimal places.

### *Example*

```
<variable ident="1" type="single">
    <name>Q2</name>
    <label>How did you like the taste?</label>
    <position start="3" />
    <values>
        <value code="1" score="2">Liked it very much</value>
        <value code="2" score="1">Liked it a little</value>
        <value code="3" score="0">Neither liked not disliked it</value>
        <value code="4" score="-1">Disliked it a little </value>
        <value code="5" score="-2">Disliked it very much </value>
        <value code="9">DK/NS</value>
    </values>
</variable>
```

### *Features and Benefits*

Enables score values for the purpose of computing statistics, assigned by the person carrying out the survey or carrying out initial analysis, to be transferred to another analysis program, or archived.

### *Notes*

The omission of a score implies that records having that value code should be omitted from the base for statistical computation.

## Date/Time Values

**Provides a mechanism for defining data values that represent dates or times.**

### *Outline*

Hitherto there has been no way to indicate that a data value represents a date or time. The new version of the standard extends the variable *type* attribute so that we can have *type="date"* or *type="time"*.

The standard Triple-S data representation will be the basic ISO 8601 all-numeric date/time format. This means YYYYMMDD for dates, and HHMMSS for times.

### *Examples*

The following example shows some additional fields that are defined with the new date/time values: -

```
<variable ident="7" type="date">
    <name>DateWhen</name>
    <label>Date of visit</label>
    <position start="47" finish="54" />
    <values>
        <range from="20040101" to="20041231">
    </values>
</variable>

<variable ident="8" type="time">
    <name>TimeWhen</name>
    <label>Time of visit</label>
    <position start="55" finish="60"/>
</variable>
```

With a data file like: -

```
. . .20040823100000                 < 23rd Aug 2004, at 10.00am >
. . .20040901141500                 < 1st Sep 2004, at 2.15pm >
```

### *Features and Benefits*

This extension will allow those exporters and importers that provide explicit date/time variables to transfer them.

### *Notes*

We only handle dates and times as separate variables – there is no combined *datetime* variable type.

The choice of new variable types as the carrier for date/time values has some implications for importing systems that do not support dates or times. They can choose to ignore these variables, or store them as characters or quantities. However, arithmetic (e.g. calculating means) on any underlying Quantity variables should be avoided.

As the data size and representation for a date or time is known (8 characters for a date, 6 for a time) the exporter need not provide a *<values>* block. If there is a *<values>* block then it must contain a *<range>* element to indicate the range of valid dates or times, and/or explicit *<value>* elements to define special dates/times.

Exporters can still output different formats for date/time data values by using Character variables. However, only the format described here can allow an importer to automatically handle date/time values

## Specialised Texts

**Triple-S XML 1.2 included an implementation of multilingual texts. At version 2.0, we introduce specialised or modal texts.**

### *Outline*

Specialised or modal texts enable alternative texts to be specified for different uses such as for interviewing and for analysis. This is done by adding a `mode=` attribute to the <text> element introduced in Triple-S XML 1.2.

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.

### *Examples*

The more of an item of text is specified using a mode attribute and corresponding modes list attribute as in the following example:

```
<sss version="2.0" modes="interview analysis">
 ...
<variable ident="1" type="quantity">
  <name>Q1</name>
  <label>Age
    <text mode="interview">How old are you?</text>
    <text mode="analysis">Age of respondent?</text>
  ...
```

The same principle is be extended to include a mode attribute and corresponding modes list attribute as in the following example:

```
<sss version="2.0" languages="en-GB fr" modes="interview analysis">
 ...
<variable ident="1" type="quantity">
  <name>Q1</name>
  <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>
  ...
```

### *Notes*

Where mode and/or language specialisations are used it is recommended that the outermost text string is always present, thus providing a default string for importers who neither distinguish language or mode.

## Limits and Restrictions

**Removes and revises restrictions on elements and attributes to make Triple-S easier to use.**

### *Outline*

Till now, the Triple-S definition has contained very few limits and restrictions. Whilst being very laudable in providing freedom for exporters, this can make life unnecessarily difficult for importers. We have now introduced some limits and restrictions.

### *Changes*

- We have removed the 4-digit limit on the *ident* attribute in the *<variable>* element. This is a legacy restriction that has no current use. The value should now be any positive numeric integer with optional leading zeros:-

  ```
  <variable ident="090001" type="single">
  ```

- We have removed the optional attribute *options=standardnames* in the *<sss>* element. This has not proved useful. In general, importers cannot always be sure that the variable names do in fact conform to the definition of Triple-S standard names. Even if they do, then the definition is probably too restrictive, and possibly not appropriate to their own conventions.
- All variable names must be unique within the *<record>* block. In practice a name is not useful if it can be re-used within the record. The check for uniqueness will be case-sensitive, and will ignore any leading or trailing blanks.
- The use of *names* (survey, variable and level) is made easier by restricting them to a useful minimal subset of the definition for an XML name:-

  The name must begin with a letter (A-Z or a-z), or _ (underscore) character.

  Subsequent characters can be letters (A-Z and a-z), digits (0-9), . (period), or _ (underscore).

  ```
  <variable ident="91" type="multiple">
     <name>Q17.1</name>
  ```

- All numbers used as attribute values should be restricted to 32-bit integer values (i.e. -2147483648 to 2147483647). This limit affects at least the *ident* attribute in a *<variable>* element, locations within a *<position>* element, values within a *<range>* element, and a numeric *code* and *score* within a *<value>* element.

### *Features and Benefits*

Changing the limits and restrictions should make the Triple-S specification more consistent, and make it easier for importers to cope with the potentially large variation in exports.

### *Notes*

The restrictions on variable names imply that blank (i.e. empty) *<name>* elements are no longer valid.

The use of hierarchical data imposes additional uniqueness restrictions on variable names.

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).