Triple-S

A standard within a standard

Keith Hughes, Stephen Jenkins and Geoff Wright

March 1999

Abstract

The Triple-S standard defines a means by which both survey data and variables may be transferred between different survey programs running on different software and hardware platforms. As such it represents a first real attempt to resolve what should be a simple, but was previously a complex, process. This paper gives a brief history of the development of the Triple-S standard, outlines the formal aims of the original specification and introduces the key aspects of the published standard and the revised version (1.1) published in 1998. We also look at the development of the internet as a communication mechanism and the recent rise in the use of XML as a means of describing and categorising the content of documents. We show how this aspect of XML leads to its logical choice as a meta standard for describing survey information. We move on to describe the contents of the proposed Triple-S XML standard (which is fundamentally the current Triple-S standard expressed in XML syntax); the benefits of extensibility that XML structure brings and the significance that the move to XML brings, for both software developers and users alike.

Keywords: Survey interchange; survey transfer; interchange standard; survey data; survey metadata

1. Introduction

Increasingly, users of survey software are demanding that data be exchangeable between survey software systems from different vendors and possibly running on different hardware and/or different operating system platforms. The transfer may be required because an client wants to perform some more detailed analysis of aspects of a survey originally conducted by an agency and the two parties use different survey software. Or the transfer may be required between a software program specialising in part of the survey process, such as CATI or document (questionnaire) scanning, and one specialising in another part, such as results analysis.

The transfer of data between programs is usually a fairly straightforward process, provided that both the exporting and importing programs can agree on an appropriate, usually ASCII text based, format. However, the transfer of survey variables is often not so simple.

This is the background which led us to write the Triple-S survey interchange standard and which has subsequently been adopted by a number of survey software packages worldwide.

Since the time the standard was originally published, developments in communication have taken place at such a pace that the transfer of data and meta-data between *computers* is common and relatively simple using Internet based technologies. By adapting the Triple-S standard to one of the standard Internet languages, XML, we are ensuring that Triple-S is able to grow and evolve from its base as the standard for survey interchange.

2. The background of Triple-S

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 original version 1.0 was devised by the present authors and published in 1994. Over 100 copies of the specification have been requested by, and sent to software developers and user organisations around the world. Since then, and partially as a result of feedback from those interested parties, the original scope was extended and published in 1998 as Triple-S 1.1.

Triple-S has been designed as an *interchange* format. It was not conceived as a native survey definition format, nor is it a replacement for the many proprietary survey definition languages currently in use. Provided that the chosen software supports the Triple-S standard, surveys can be ported between different hardware and software platforms using Triple-S format files on any convenient medium including diskettes, magnetic tape and so on.

The primary design goals for the standard were that:

- It encapsulates the definitions and data of a large number of common surveys.
- It is reasonably simple to implement for both export and import.
- It does not prejudice further development by incorporating more than is needed. It was never an intention to make this the first and last attempt but rather to make it a good basis for further development with the benefit of experiences of both users and implementers.

In practice, when attempting to devise such a standard, there are inevitably some compromises to be made. We wanted to produce a standard in a reasonable time scale and produce one which ‘works’ in the sense that software users and developers would use it. We also wanted to avoid adopting a ‘lowest common denominator’ solution to the problems raised. Triple-S therefore had to deal with realistically large problem sizes.

The approach adopted was to use ASCII text based files. A text based layout avoids the problems associated with transferring binary files - clearly this is an important point since these files are *designed* to be moved and transferred between different computer systems. The following additional benefits were perceived:

1. The files can be inspected using a suitable text editor which speeds the development of both import and export programs.
2. Import functions could be written before equivalent export functions (if any) by hand-coding suitable Triple-S files.
3. Users could introduce corrections if particular implementation problems arose ‘in the field’.

It was important that the syntax used would make it a relatively simple job to write a suitable export module and import parser. An important consideration was that anyone able to write an import module should also be able to write an equivalent export module thus placing no obstacles in the way of implementing both (where appropriate) rather than just implementing an import module - we certainly did not want to see a multitude of programs importing Triple-S files but none offering an equivalent export.

It was essential to include a description of both the survey data and the survey variables as well - after all, the inability to transfer variables is what makes many of the ‘standard’ techniques, described previously, cumbersome to use.

3. The development of XML

XML is an acronym for Extensible Markup Language. It was devised by members of the World Wide Web Consortium (W3C) and its specification published in February 1998. It is a subset of the more general SGML (Standard, Generalised Markup Language).

XML can be thought of as a configurable version of the ubiquitous HTML (Hyper-text Markup Language). HTML documents, which proliferate on the web at the present time, contain only formatting information for the text and graphics they incorporate. Many people have had the experience of using web-based search engines and either getting "No documents match the request" or a message along the lines of "Approximately 1,956,281 documents match the request" and with no apparent way of bridging that gap. XML was introduced partly to resolve issues

like that – by making documents self-describing it is possible to have them respond to context specific requests. Thus instead of searching for 'Smith' and getting overwhelmed with the number of 'hits', it is possible to search XML documents for, say, Author = 'Smith' and receive a list of those only where 'Smith' was found in the 'Author' context.

Although the historical roots of XML are in document publishing, it is also suited to the more general task of unambiguously describing complex data structures that may never be viewed or printed. Because XML is used to describe information as well as structure it, it can be thought of as a data description language. Or, in the use we put it to here, as a survey description language.

Like an HTML document, an XML document contains instructions, or *tags*, which enclose identifiable parts. However, unlike HTML, the tags may be associated with application specific meaning – that is they may be ascribed some particular semantics in a particular domain.

So, whereas in HTML we may see formatting related tags such as:

```
<FONT FACE="Arial">When was your business founded?</FONT>
```

XML gives us the ability to describe tags which describe the context of the text and, in doing so, the *structure*of an object. For example:

```
<QUESTION>When was your business founded?</QUESTION>
```

Any program which reads an XML document needs some way of checking its correctness. That is, to check that the document conforms to the appropriate rules. There are two levels at which this is done:

- At the superficial level, a document may be determined to be 'well formed'. It is well formed if the tags encountered within match up and are not inter-nested, for example if for each `<QUESTION>` there is a corresponding `</QUESTION>`.
- At a more detailed level a document may be considered 'valid' if it conforms to rules specified in an appropriate DTD (Document Type Definition) file. A DTD describes the allowable tags and how they may be used in combination with one another. Thus, although the example above using the `<QUESTION>`and `</QUESTION>` tags is well formed, it will only be valid if the associated DTD contains an appropriate definition for question tags. Note that the DTD only provides for syntax checking of a document – it does not provide for any form of semantic checking.

4. Overview of Triple-S XML

The Triple-S XML format provides for the cross-platform transfer of both survey data and survey variables using universal industry standard protocols. The syntax of a Triple-S XML document is described by the freely available Triple-S DTD.

Comprehensive variable types

Triple-S XML provides for the description of the five most common types of variable:

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

Flexible data formats

Triple-S XML allows for both integer and real coded values to be represented. Two formats for the representation of multi-response data are supported.

Coding schemes are maintained through the transfer process

Where standard coding has been used to represent special values – for example, where '9' is used to represent 'Not Answered' – that coding is maintained through the transfer operation rather than being closed down on a question by question basis. Furthermore, the fact that a particular code *is* 'special' in some way can be represented and thus indicated to a survey importer.

A solution to the problem of name incompatibility.

Different survey software packages have different constraints as far as variable names are concerned. Some allow underscore characters within names, others do not. Some allow period characters ('.'); others do not. What is needed is some way of translating names from one system into equivalent names on another. Triple-S XML provides a solution to this issue by introducing the notion of *Standard Names*. In Triple-S XML, a Standard (variable) Name is one which:

1. consists of 1-8 characters
2. consists of only the characters A-Z (and a-z) and 0-9
3. starts with a letter (A-Z or a-z)
4. is case insensitive (i.e. upper and lower case are equivalent)
5. considers all characters are significant (e.g. Q001 is distinct from Q1)
6. is unique within the survey

Support tools are available for writing software

XML compatible browsers are now coming in to the market and may be used by software developers to assist with reading and writing Triple-S XML files.

5. Triple-S XML syntax and semantics

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

The syntax of a Triple-S XML Description File accords with XML guidelines. The content corresponds with the Triple-S version 1.1 specification. To aid the process of development of import/export software, both files are relatively simple to read by eye.

```
<!DOCTYPE SSS SYSTEM "c:\sss.dtd">

<SSS VERSION="1.2">                                          [1]

<DATE>2/MAR/1999</DATE>                                      [2]

<TIME>18:32</TIME>

<ORIGIN>Prog v1.42</ORIGIN>

<USER>Ann Other</USER>                                       [3]

<OPTION>STANDARD NAMES</OPTION>

                                                             [4]

<SURVEY>

                <TITLE>Historic House Exit Survey</TITLE>    [5]

                <RECORD ID="V">

                                …                            [6]

                                details for each survey
                                variable

                                …

                </RECORD>

</SURVEY>

</SSS>
```

Figure 5.1 - General layout of a Triple-S XML Definition file

The Definition file consists of a combination of XML markup tags and content definitions. The file begins [1] with an optional reference to the Triple-S DTD (Document Type Definition) which describes the allowable Triple-S syntax. This

particular example shows the DTD as being held in an external file but it may be incorporated directly or referenced from an Internet location.

The SSS definition proper begins with the SSS tag [2] which also specifies a version number to help interpret the remainder of the file. That is followed by optional date, time, origin and user specifications [3] which together provide a versioning system for the file contents. The origin is normally recorded as the name and version of the program generating the file (i.e. the program exporting the survey).

Next is an optional specification that the names of the variables recorded in the file conform to the Triple-S notion of STANDARD NAMES (outlined in section 4).

The survey itself is encapsulated using a SURVEY tag [5] which introduces the survey title and the RECORD block tag [6] which specifies an identifier. Only one RECORD block is allowed but more may be allowed in later versions to model hierarchical and other data structures. The survey variables are described within the RECORD block. Triple-S XML allows for the definition of 5 different types of variable.

The exact syntax varies by type although each begins with the same preamble. The following text highlights key points relevant to variables of each type.

All variables begin with the same sequence which is introduced by the VARIABLE tag [7] and which includes the specification of a unique, numeric identifier. This may be used by the importing program along with the corresponding RECORD identifier to generate names for variables whose real names would otherwise be illegal.

	<VARIABLE ID="1">		
	<NAME>Q1</NAME>	[7]	
	<LABEL>Number of visits</LABEL>	[8]	
	<TYPE>SINGLE</TYPE>		
	<POSITION>1</POSITION>	[9]	
	<VALUES>		
	<VALUE CODE="1">First visit</VALUE>		
	<VALUE CODE="2">Visited before within the year</VALUE>	[10]	
	<VALUE CODE="3">Visited before that</VALUE>		
	</VALUES>		
	</VARIABLE>		

Figure 5.2 - A typical SINGLE type variable

Definition of the variable itself then begins with the NAME tag [8] (the contents of which would conform to the Triple-S standard definition if the option STANDARD NAMES was included) and a textual LABEL. The TYPE is specified next [9] and would be expected to be one of the five intrinsic types of SINGLE, MULTIPLE, CHARACTER, QUANTITY or LOGICAL. Following the type is the POSITION specification which describes the location of the data values within the data record either as a single value (as shown) or as a range (as shown in Figure 5.3). The remaining definition varies according to the type.

SINGLE type variables (Figure 5.2) are completed by the specification of a VALUES block which is composed from a number of contained VALUE definitions [10]. Each VALUE definition describes a CODE and a LABEL. The CODE is an integer number and represents the data (at the specified POSITION) corresponding to the associated LABEL.

Variables of type MULTIPLE (refer to Figure 5.3) begin with a similar preamble to those of type SINGLE. Both include NAME, LABEL, TYPE [11] and POSITION tags but beyond that, the MULTIPLE example shown includes a SPREAD tag [12].

```
<VARIABLE ID="4">
<NAME>Q4</NAME>
<LABEL>Two favourite attractions visited</LABEL>                    [11]
<TYPE>MULTIPLE</TYPE>
<POSITION>41 TO 42</POSITION>
<SPREAD>2</SPREAD>
<VALUES>                                                            [12]
<VALUE CODE="1">Sherwood Forest</VALUE>
<VALUE CODE="2">Nottingham Castle</VALUE>
<VALUE CODE="3">&quot;Friar's&quot; Restaurant</VALUE>
<VALUE CODE="4">&quot;Maid Marion&quot; Cafe</VALUE>
<VALUE CODE="5">Mining museum</VALUE>
<VALUE CODE="9">Other</VALUE>
</VALUES>                                                           [13]
</VARIABLE>
```

Figure 5.3 - A typical MULTIPLE type variable

The SPREAD tag [12] is used to direct a new feature introduced in Triple-S 1.1, and retained for Triple-S XML, which is the ability to specify that data for a Multiple is recorded as either one character per value (bitstring format), or in SPREAD format where the same information is coded in contiguous sub-fields.

The SPREAD format has the benefits that both the original coding scheme and the order of mention are preserved. It is also more space-efficient if the respondent is not allowed to select all of the possible replies. For example, if a respondent chose "Sherwood Forest" (code 1) and "Mining museum" (code 5) at the variable in Figure 5.3, they would have data recorded as:

15

By not mentioning SPREAD, the data file would be recorded as a bitstring:

100010000

Notice that the bitstring equivalent is 9 characters long to allow for the code "9" to represent "Other" as per the specification [13].

Character type variables (Figure 5.4) only have a SIZE declaration [14] in addition to the standard peamble. The SIZE simply specifies the maximum number of characters that will be expected in the data file. There is no way of specifying variable-length character fields or of specifying valid ranges or individual values.

	<VARIABLE ID="3"> <NAME>Q3</NAME> <LABEL> Other attractions visited </LABEL> <TYPE>CHARACTER</TYPE> <POSITION>11 TO 40</POSITION> <SIZE>30</SIZE> </VARIABLE>	[14]	

Figure 5.4 - A typical CHARACTER type variable

Quantity type variables (Figure 5.5) on the other hand provide for much more detailed specification. In addition to the standard preamble, a VALUES block provides for the optional inclusion of a RANGE specification [15] and an optional WITH clause identifying discrete values.

```
<VARIABLE ID="5">
<NAME>Q5</NAME>
<LABEL>Miles travelled</LABEL>
<TYPE>QUANTITY</TYPE>
<POSITION>43 TO 45</POSITION>
<VALUES>
<RANGE>1 TO 499</RANGE>                                    [15]
<WITH>                                                     [16]
<VALUE CODE="500">500 or more</VALUE>
<VALUE CODE="999">Not stated</VALUE><SPECIAL/>             [17]
</WITH>
</VALUES>
</VARIABLE>
```

Figure 5.5 - A typical QUANTITY type variable

The start value and finish value located between the RANGE tags [15] explicitly define the valid range and implicitly define the format and physical size of data for the variable. The valid range for a Quantity type variable may include positive or negative, integer or 'real' (decimal) values.

The WITH clause [16] provides an opportunity to label some of the values explicitly. The labeled values may lie within, or beyond the range described by the RANGE specification. The optional SPECIAL marker [17] may be used to indicate that the value may be treated in some special way (e.g. missing data, don't know etc.).

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

```
<VARIABLE ID="6">
<NAME>Q6</NAME>
<LABEL>Enjoyed visit</LABEL>
<TYPE>LOGICAL</TYPE>
<POSITION>46</POSITION>                                    [18]
```

```
</VARIABLE>
```

Figure 5.6 - A typical LOGICAL type variable

Finally, variables of type LOGICAL have no additional information recorded other than being identified as of that type [18]. Data for LOGICAL variables only ever occupies a single position.

6. Comparing Triple-S XML with Triple-S v1.1

As has been mentioned previously, the conversion of Triple-S Definition Files to an XML based syntax is relatively straightforward because of the similarity of the structures and scope of the two forms.

Here we compare the two by considering the specification of a multiple type variable in each of the specification formats:

```
<VARIABLE ID="4">
<NAME>Q4</NAME>                                                    [1]
<LABEL>Two favourite attractions visited</LABEL>
<TYPE>MULTIPLE</TYPE>
<POSITION>41 TO 42</POSITION>
<SPREAD>2</SPREAD>                                                 [2]
<VALUES>
<VALUE CODE="1">Sherwood Forest</VALUE>
<VALUE CODE="2">Nottingham Castle</VALUE>                          [3]
<VALUE CODE="3">&quot;Friar Tuck&quot; Restaurant</VALUE>
<VALUE CODE="4">&quot;Maid Marion&quot; Cafe</VALUE>                [4]
<VALUE CODE="5">Mining museum</VALUE>
<VALUE CODE="9">Other</VALUE>
</VALUES>
</VARIABLE>
```

Figure 6.1 - Sample variable in Triple-S XML

```
VARIABLE 4
NAME "Q4"                                                          [1]
```

	LABEL "Two favourite attractions visited" TYPE MULTIPLE POSITION 41 TO 42 SPREAD 2	[2]	
	VALUES 1 "Sherwood Forest" 2 "Nottingham Castle"	[3]	
	3 "{"}Friar Tuck{"} Restaurant" 4 "{"}Maid Marion{"} Cafe"	[4]	
	5 "Mining museum" 9 "Other" END VALUES END VARIABLE		

Figure 6.1 - Sample variable in Triple-S 1.1

As can be seen, the structure of both samples is very similar. The following points are worthy of note:

- The variable identity is expressed as an attribute of the variable element in the XML version [1]. In the second example (Triple-S 1.1), it is identified in a similar although not such a specific way.
- The POSITION specification [2], of the XML version would return the string "41 TO 42" to the reading program whereas the Triple-S 1.1 version would return the two numbers as separate items (having written a suitable parser of course!). However it is a simple matter in most languages to search for a substring within a character string. Using such a facility, if "TO" is found then the two numbers may be picked of from either side; if not found then assume that the specification is for a single position.
- The specification of each VALUE [3] has an explicit CODE identifier in the XML version whereas the code is identified by position only in Triple-S 1.1.
- XML has a pre-specified way of introducing control characters into texts. For example quotation marks are represented as ". Similar substitutions would be made for "<" (<), ">" (>) and ampersand, "&" (&). Triple-S used curly braces, { and }, to surround special characters within a text string.

In summary, the format of the survey data file remains unchanged as does its relationship with the corresponding definition file. Furthermore, although the

syntax of the declarations in the definition has (necessarily) changed to conform to XML standards, the structure adheres closely to the existing Triple-S standard. The impact on both users and software developers has thus been minimised – software authors will obviously need to do some work to adapt existing Triple-S implementations but software users should detect no operational differences at all.

7. Other work

We believe that Triple-S XML provides a unique solution to the problem of exchanging survey data between otherwise incompatible survey software packages but it is not alone in using XML to represent survey variables.

In 1995, the Inter-university Consortium for Political and Social Research (ICSPR), established the Data Documentation Initiative to develop a Document Type Definition (DTD) for an international codebook standard using SGML. In 1997, the DTD was made compliant with XML (Extensible Markup Language). The codebook provides for the definition of survey variables and corresponding data although it differs from Triple-S in its aims and thus its principle features.

The DDI provides a means of documenting past surveys for collation in a repository of such surveys, in particular:

- It provides facilities to describe the storage of survey data rather than prescribing a method. From the perspective of constructing an archive of survey data therefore, the DDI works well in that it provides a great deal of flexibility in the data that is deemed acceptable.
- It provides for very detailed textual annotation of surveys and survey variables including details of question phrasing and instructions to coders. This is clearly of benefit for those wanting to understand the specific semantics of individual questions and reflects the main requirement of constructing a well-documented archive.

Both the above aspects portray the difference in focus between the DDI and Triples-S XML. Our goal with Triple-S XML is to provide a description of most surveys that is easy to implement and provides an automated solution to the problem of survey interchange.

8. Future directions

At the time of writing (March 1999) the specification of Triple-S XML described is towards the end of the discussion phase. We do not expect to make any significant changes in scope of the standard at this stage although detailed syntactical changes (within the confines of the XML meta-standard) may be introduced before final publication.

Looking beyond that stage, a number of possible enhancements are being considered:

- Incorporating inverted data or summary data for each variable. By this we mean that the data for any particular variable would also be represented in the same file as the variable itself. A feature such as this would make it easier to transfer a small number of variables out of the context of a complete survey, rather than transferring the entire survey. It would also mean that we could consider some, possibly more efficient, 'internal' format of data transferred in this way but still leave the current method (where data is held in a separate file) for transfer of 'raw' data.
- Allow the description of structured (hierarchical) data. This feature would enable at least two-level hierarchical datasets to be described. This would be sufficient for representing simple household / person surveys for example. However, the scope could be extended much further to hierarchies with more than two levels (for example, Household / Person / Trip structures) or, indeed, to the ability to describe any dataset which can be represented by a directed acyclic graph.
- Incorporating calculated fields and derived values. The issue here is that different software packages use different (although often very similar) languages for expressing the same formulaic constructs. However, it may be possible to use MathML (a mathematical language, also based on XML) to provide a common framework for describing derivations and other calculated values.

By extending the scope sufficiently, it may be possible to use Triple-S XML as the basis for a more general-purpose, standard SurveyML. We are quite excited by the opportunities this would offer.

9. The Triple-S XML DTD

The proposed Triple-S XML 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'.

```
<?XML version="1.0" ?><!DOCTYPE SSS [ <!ELEMENT main SSS>

<!ELEMENT SSS (date, time, origin, version, options?, survey)>

<!ATTLIST SSS version (CDATA)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT time (#PCDATA)>
```

```
<!ELEMENT origin (#PCDATA)>

<!ELEMENT user (#PCDATA)>

<!ELEMENT options ("STANDARD NAMES")>

<!ELEMENT survey (title, record)

<!ELEMENT title (#PCDATA)>

<!ELEMENT record (variable+)>

<!ATTLIST record id CDATA #REQUIRED>

<!ELEMENT variable (name, label, type, position, (values | size)?)>

<!ATTLIST variable id CDATA #REQUIRED> <!ELEMENT name (#PCDATA)> <!ELEMENT label (#PCDATA)>
<!ELEMENT type ("SINGLE"|"MULTIPLE"|"CHARACTER"|"QUANTITY"|"LOGICAL")>

<!ELEMENT position (#PCDATA)> <!ELEMENT values ((value, special?)+ | (range?, with?))>

<!ELEMENT value (#PCDATA)>

<!ATTLIST value code CDATA #IMPLIED>

<!ELEMENT range (#PCDATA)>

<!ELEMENT with ((value, special?)+)>

<!ELEMENT special EMPTY>

<!ELEMENT size (CDATA)>

]>
```

10. References

(1) Wills, P. (1992), "Data Use and Reuse: The movement and use of survey data across different hardware and software platforms", Survey and Statistical Computing 1992, pp297-304.

(2) Jenkins, S. (1996), "The Triple-S Survey Interchange Standard: The story so far" , Survey and Statistical Computing 1996, pp341-350.

(3) Hughes, K., Jenkins, S. and Wright, G. (1998), "The Triple-S Survey Interchange Standard, version 1.1", http://www.asc.org.uk/SSS/index.htm.

(4) Bradley, N. (1998), "The XML Companion", Addison Wesley Longman.

(5) ICSPR "Data Documentation Initiative" http://www.icpsr.umich.edu/DDI

11. About the authors

Keith Hughes is Development Director of Merlinco Ltd. He has been involved in the development of survey software on all kinds of system platforms for over 30 years. He can be

contacted at Merlinco Ltd., 33 Welbeck Street, London W1G 8EX, England; tel +44 (0)171 486 6229; fax +44 (0)171 486 4322; e-mail:merlincoltd@compuserve.com

Stephen Jenkins is Technical Director of Mercator Computer Systems Ltd. He has over 25 years experience in designing and writing software for survey work. He designs and manages development of the snap research analysis software. He can be contacted at Mercator Computer Systems Ltd., 5 Mead Court, Thornbury, Bristol BS35 3UW, England; tel +44 (0)1454 281211; fax +44 (0)1454 281216; e-mail sjenkins@mercator.co.uk

Geoff Wright is Senior Software Developer with The Arbitron Company, having formerly held that position with Pulse Train Technology Ltd, authors of the Bellview and STAR survey software. He can be contacted at Arbitron UK, Enigma House, 30 Alan Turing Road, Guildford GU2 5AA, England; tel +44(0)1483 251302; e-mail geoff_42@hotmail.com