RAL-93-029

Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-93-029

SGML Tables for the PHIGS Slide Set

R E Thomas

May 1993

SGML Tables
for the
PHIGS Slide Set

R E Thomas

1 May 1993

SGML Tables for the PHIGS Slide Set

1. INTRODUCTION

The paper "SGML in Practice, The PHIGS Slide Set" (RAL-92-071) describes the work undertaken to convert the text of a large set of slides to make use of the Standard Generalized Markup Language (SGML). This made it much easier for those in Higher Educational Institutions (HEIs) to understand the semantics of each slide, and thus incorporate selected parts into existing on-line slide sets without introducing factual errors. The version described, however, did not attempt to provide markup for the tabular information included in the slides. Instead, the original tables (in tbl format, part of the Documenter's Workbench[1] tool set) were retained, with tags inserted before and after each table.

This paper discusses the problems related to tabular information in general, and the particular tables used in the Slide Set. A suitable Document Type Definition (DTD) fragment for the Slide Set, which captures the essential semantics, is described.

2. BACKGROUND

Users of SGML have always had problems with the provision of markup for tables, and there are still discussions at International level on the best way of tackling this. In addition, some of the decisions taken in the initial work on the Slide Set have introduced problems which need to be solved if the resulting DTD is to be generally useful. In the context of the Slide Set, it is important to be able to distinguish those features of the layout which "do not matter" from those features which are an essential component (ie the table would convey a different meaning if the feature was altered). Of course, the physical relationship of the data itself is recognised by everyone as essential.

2.1 General Problems

The main difficulty is to separate the essential semantics of a table from its visual layout. At one extreme, a table can be considered merely as a rectangular array of data, and everything else pure presentation. Another, opposite view, says that the whole layout is an important part of the table and cannot be divorced from the data within. A better view will lie between the two.

For example, placing a box round a table (ie lines top and bottom, and down both sides) makes the table stand out, but does not add anything to the way the contents of the table is perceived. It is thus a form of highlighting, and can be treated as such. Similarly, positioning the whole table in the centre of a page is another form of highlight (see below).

Unboxed Left-Justified Table

Entry1	20	Entry3
Entry4	30	Entry6
Entry7	40	Entry9

1. Documenter's Workbench is a registered trademark of AT&T in the USA and other countries

Boxed Centred Table

Entry1	20	Entry3
Entry4	30	Entry6
Entry7	40	Entry9

On the other hand, puting vertical lines between some columns and not others introduces a grouping which has semantic relevance. Changing such lines changes the way the data is interpreted. Thus the position of these lines contains semantic information and must be captured in any markup scheme which is intended to preserve meaning (compare the two tables below).

Entry1	20	Entry3
Entry4	30	Entry6
Entry7	40	Entry9

Entry1	20	Entry3
Entry4	30	Entry6
Entry7	40	Entry9

Note that the following discussion and "solution" does not attempt to solve the general problem. Indeed, it is recognised that there are many forms of table layout which none of the existing DTDs can handle (eg a table in which a sub-array of elements is enclosed in its own box, independent of the row and column separators).

2.2 Specific Problems

At the start of the conversion to SGML, it was decided to cut down on the number of tags generated. Thus a new tag was not defined if an existing tag could produce the same visible output. It is possible to interpret a particular layout in more than one way: for example, a single column of numbers might be a "simple list" or a "one-column table". A two-column table where the first column contains sequential integers might be an "ordered list" (see below).

Entry1
Entry4
Entry7

1 EntryA
2 EntryB
3 EntryC

In each case, the 'table' interpretation was adopted, without considering whether lists should be identified explicitly.

The other major problem was the decision to present the table contents in tbl form. There is an obvious difficulty in providing a suitable interpretation of the table contents for those who do not know tbl. A better method of presentation is required.

3. ANALYSIS OF TABLES

It is important that the chosen markup is capable of reproducing the existing slide set. It is therefore necessary to analyse the tables that are currently used to provide guidance on the tags and attributes needed. However,the tables were created by different authors at different times, with no attempt at consistency, so the result of the analysis may lead to a need to change the original source, if a better solution can be found.

As stated earlier, some of the visual aspects of a table serve merely to highlight the table, rather than contribute to the semantic content. Within the PHIGS slide set, there are three features which come under this heading: centring of the whole table, putting a box round the whole table, and selecting a larger font for the whole table. These features are used separately or together. Since they do not affect the semantics, the method of indication should either be by use of a "style" attribute on the main table tag, or separate highlight tags which can enclose the whole table section. This second solution would be in keeping with other highlight tags which indicate bold or italic phrases. Indeed, in the case of centring and large characters, such tags could have meaning in contexts other than tables. These features will not be considered further when identifying the different types of table used, being equally applicable to all.

Each table appearing in the slide set was classified according to a collection of distinguishing features. The number of classes was kept to a minimum. Of course, many different classification schemes are possible, but the list below covered all the examples available. Apart from the List type, the number of columns in the table is not seen as a class distinguishing feature.

Lists
No Headings
Single Line Column Headings
Multiline Column Headings
Spanned Headings
Full Spanned Headings and Subheadings
Partial Spanned Headings and Subheadings
Complex

Each will be considered separately and the main markup issues identified.

3.1 Lists

As described in a previous section, it is possible to treat a single column table as a simple

list, and a two column table whose first column is a sequence of integers as an ordered list. At present, they are actually treated in the same way as Simple Tables.

3.2 Simple Tables

An examples is given below.

```
Entry1   EntryA   EntryX
Entry2   EntryB   EntryY
Entry3   EntryC   EntryZ
```

These are really arrays of data without headings or separators (horizontal or vertical lines between rows and columns). They are very common, and the chosen markup should reflect the simplicity (eg the default settings for attributes should give rise to this class of table).

3.3 Single Line Column Headings

This also appears frequently in the slide set. Vertical separators appear between each column, and a horizontal separator is used to differentiate between the heading and the data. The appearance of the vertical separators introduces an additional feature: the relationship of the data to the column edge. Three types of alignment are used: all centred, all left justified, mixed justification (see examples below).

H1	H2
Entry1 Entry2 Entry3	EntryA EntryB EntryC

H1	H2
Entry1 Entry2 Entry3	EntryA EntryB EntryC

H1	H2
Entry1 Entry2 Entry3	EntryA EntryB EntryC

The vertical alignment certainly changes the look of the table, and the author has been unable to find an example where such variations in alignment have semantic content. It is usual to give some indication of the author's layout wishes by indicating alignment as an attribute (at various levels, from the whole table to an individual cell). Separate alignments for title and data are common.

3.4 Multiline Column Headings

Two forms of multiline headings are used in the slide set:

H1 H1.1	H2 H2.1
Entry1 Entry2 Entry3	EntryA EntryB EntryC

H1 H1.1	H2 H2.1
Entry1	EntryA
Entry2	EntryB
Entry3	EntryC

In the first example, the two lines of heading could equally be combined into one if the columns were wider (see below), in which case this could be treated as another example of the previous type (letting the chosen formatter decide whether to split the heading or not). As an illustration, consider the following (taken from the original Slide Set).

Structure	Pick Identifier	Element Position
WKNV	0	2
DESK	0	1
LG	0	3
BLOT	0	1

Structure	Pick Identifier	Element Position
WKNV	0	2
DESK	0	1
LG	0	3
BLOT	0	1

It is more usual however, to indicate multiple rows explicitly (by tag): either as a set of heading rows or as heading and subheading.

The second example includes row separators. Used regularly in this way, they do not provide any additional semantic information, and could be treated as an attribute (eg on a row). Some systems however use an empty tag to represent the separator explicitly.

3.5 Spanned Headings

There is one case of a table which has a single line heading spanning the whole width of the table (not confined to a particular column). Its alignment is specified in relation to the full table.

```
                Head1
        ------------------------
        Entry1      EntryA
        Entry2      EntryB
        Entry3      EntryC
```

It is possible to consider this as a special case of either of the next two classes (full or partial spanned headings with subheadings). On the other hand, it may be more appropriate to consider such a full-width heading as being associated with the table as a whole rather than with a collection of columns. This introduces either the idea of a separate table title (with possibly distinct tags) or a caption (which commonly appears as a form of labelling device for the table, outside any structure such as a containing box). The current slide set does not contain any other example of a specific caption.

To date, this class has been treated as a special case of later classes. The span does have semantic significance (changing the span would change the meaning). Several different methods of indicating the size of the span have been tried in various DTDs (see later).

3.6 Full Spanned Headings and Subheadings

The more usual form of spanned heading includes additional lines which do correspond to particular columns. Examples are given below.

Head1	
H1.1	H2.1
Entry1 Entry2 Entry3	EntryA EntryB EntryC

Head1		
H1.1	H2.1	H3.1
Entry1	EntryA	EntryX
Entry2	EntryB	EntryY
Entry3	EntryC	EntryZ

In the first example, the spanned heading bears a close relation to the subheading (this relation is emphasised by the placing of the row separator) and this would seem to be a clear case where the "heading and subheading" interpretation is appropriate. In the second example however, the additional row separator between the heading and subheading again raises the question as to whether a more appropriate interpretation would be to take the spanned heading as a caption or as a (separate) title. As an illustration, consider the following table (taken from the original Slide Set) and the two variations.

Polyline primitive created			
Linetype	Linewidth Scale factor	Colour Index	Polyline Index
SOLID		2	1

Polyline primitive created

Linetype	Linewidth Scale factor	Colour Index	Polyline Index
SOLID		2	1

Linetype	Linewidth Scale factor	Colour Index	Polyline Index
SOLID		2	1

Polyline primitive created

Interpreting the first line of the heading as a title would imply a separate tag for this line, and full header status (as opposed to subheading) for the second. The two elements might even be at different levels in the structure in DTDs where tags delimit header and body sections. A title element would appear between the initial table tag and the 'start of header' tag.

3.7 Partial Spanned Headings and Subheadings

Spanned headings can also be used over a subset of the columns, as in the following examples.

Head1		Head2	
H1.1	H2.1	H3.1	H4.1
Entry1	EntryA	EntryX	Entrya
Entry2	EntryB	EntryY	Entryb
Entry3	EntryC	EntryZ	Entryc

H1 H1.1	Head1	
	H2.1	H3.1
Entry1 Entry2 Entry3	EntryA EntryB EntryC	EntryX EntryY EntryZ

The first example is straightforward, but demonstrates the need for a method of indicating which columns should be spanned. Obviously the feature has semantic relevance. It is usual to indicate the span requirement by means of an attribute, but there are several ways of indicating the limits. Some systems put these as attribute values directly on the "cell" tag (ie the tag immediately preceding the actual text). Others use an empty tag to define the span, and then reference this tag in the cell entry.

The second example is more complex. The chosen tag scheme needs to be able to associate H1 and H1.1, and Head1 with both H2.1 and H3.1. This can be viewed as two lines of title with a partial row separator, a wide row with a split data cell, or possibly as some form of sub-table construct. However, the additional row separator does not seem to add any semantic information (compare with the example below):

H1 H1.1	Head1 H2.1	H3.1
Entry1 Entry2 Entry3	EntryA EntryB EntryC	EntryX EntryY EntryZ

This suggests that the sub-table approach is inappropriate. Treating this as an example of a wide row introduces a new level of complexity, including possible horizontal alignment and horizontal spanning.

It is also possible to consider the subheadings as defining a tree structure relationship between the columns. This suggests that it might be useful to investigate the presentation of such structures used in other fields. The problem with this approach is that the natural way for tabular data to be presented is by row: presentation by column leads to very long, thin source texts! It would not be helpful to have two orthogonal paradigms for the descriptions of headings and data. Consequently, this approach has not been adopted.

3.8 Complex

Finally, there is one table form which does not fit into the previous categories (only one example of this appears in the set).

Entry1	EntryA	EntryX
	EntryB	EntryY
	EntryC	EntryZ
Entry2	EntryA	EntryX
	EntryB	EntryY
	EntryC	EntryZ
Entry3	EntryA	EntryX
	EntryB	EntryY
	EntryC	EntryZ

This is the only case where row separators appear in the data in an irregular pattern. This pattern is an integral part of the table. It would be possible to tag this example by indicating that two of the rows should be followed by row separators, but this does not really capture the semantic content of the construct.

Providing an adequate interpretation, however, is difficult. It might be interpreted as a sequence of tables, where the three entities separated by the horizontal lines are considered separate. This would remove any connection between the contents of these tables, and does not offer a satisfactory solution. The alternative is to consider this to be a table where an entry in an individual cell is not just a single data item, but a construct. Thus the table in the example is interpreted as having three rows and two columns, with the cells in the second column being either ordered lists or sub-tables (depending on the form and values of EntryA, EntryB, EntryC). The row separators now make a regular pattern. This interpretation would allow a formatter to display the construct as follows (items in the first column centred in each row):

	EntryA	EntryX
Entry1	EntryB	EntryY
	EntryC	EntryZ
	EntryA	EntryX
Entry2	EntryB	EntryY
	EntryC	EntryZ
	EntryA	EntryX
Entry3	EntryB	EntryY
	EntryC	EntryZ

This could alter the meaning conveyed in some cirumstances, however. Consider for example the case where the separate sections represent quarterly data, and the entries in the first column

are the names of the first month of each quarter. The horizontal alignment would then be important.

3.9 Summary

The intention of this analysis is to identify the semantic features which are used within the slide set, so that the selected SGML definitions will correctly convey the meaning, as well as allowing a formatter to reproduce the layout of the original. Considering the examples above, the following features need to be addressed:

Lists (separate type or form of table);
Column headings identified (distinct from table data);
Multi-line headings;
Captions or Titles (needed or not);
Spanned headings (whole or in part);
Complex entries;
Row and column separators as attributes;
Heading and data alignment within columns;
Row spanning and alignment (needed or not);
Box, centring and character size as table attributes.

4. THE SOLUTION

There are a number of examples of published DTDs which have included markup for tables. It therefore seems sensible to consider whether any of these might provide a suitable base for use in the slide set (rather than defining another set of tags from scratch). Appendix 1 provides brief descriptions of the following (in alphabetical order), together (where appropriate) with example markup:

Air Transport Association (ATA);
Association of American Publishers (AAP);
Addison-Wesley;
DocBook (HaL and O'Reilley);
Exoterica (ECT);
ISO/IEC TR 9573-11;
MIL-M-28001A;
SoftQuad;
Text Encoding Initiative (TEI).

Following this analysis, it was decided, as far as possible, to select a subset of an existing table markup scheme, and apply that to the tables in the slide set. This meant that the basic characteristics of the tables had to be handled by the DTD fragment. One of these characteristics was the use of headings which spanned more than one column, and so any DTD fragment without spans was discarded (DocBook and MIL-M-28001). Another characteristic concerned the centring of the whole table and the character sizes within it. None of the fragments included markup features to handle this, although the "tabstyle" attribute in MIL-M-28001A could fulfil this role.

The other deciding factor concerned the choice of translator for the SGML. The system used was MarkIt, from Sema, running on a SUN. The translation system (an extension of the LINK section) uses fairly simple constructs. In particular, the only way it could handle token lists (such as produced by NMTOKENS or NUMBERS attribute types) was to identify whether a specific token was present or not. A string of symbols whose order was important, and where the same symbol could occur more than once, could not be interpreted. This effectively excluded all the DTD fragments apart from Addison-Wesley, MIL-M-28001A and the SoftQuad set.

Eventually, MIL-M-28001A was selected. The Addison-Wesley set does not have sufficient capability to provide the desired alignment information, although its choice of tags does provide one of the more readable solutions. The SoftQuad fragment concentrates on layout almost exclusively (creating a box around the table is a complex operation involving the specification of individual sides at different points in the table).

The points raised at the end the last chapter are considered below. The decisions taken are based on the premise that, where applicable, the simplest alternative should be selected.

a) Lists.

In the interests of simplicity, it was decided to treat possible lists as examples of tables, and not create a new set of elements.

b) Column Head Identification and Multiline Headings.

This is a feature of almost every DTD (including the MIL set).

c) Captions.

It was decided not to identify separate captions, since this did not appear to be needed in the current set of slides, where all the spanned titles can be considered headings. This means that the title element in the MIL set is not required.

d) Spanned Headings.

This is a feature of the MIL set.

e) Complex Tables.

After much discussion, it was decided to use the existing features of the MIL set to construct tables with irregular separators. The result of this decision can be seen in the example in Appendix 3. This does not really address the issue in a satisfactory way, since the true semantics of the table are not properly identified. However, as stated earlier, only one example of this type of table appears. Any improved solution would involve the creation of several new elements, and increase the complexity of the translation. For these reasons, the simple solution was adopted.

f) Row/Column Separator Attributes and Column Alignment.

This is a feature of the MIL set.

g) Row Alignment.

Although this feature is available in the MIL set, it was decided that it was not required.

h) Box, Centring, Character Size.

While there was some merit in considering the use of tags to identify highlighting, the existence of a "tabstyle" attribute in the MIL set led to its adoption as the solution. The added generality of the separate tag solution was not needed in the current set of slides.

The decisions recorded above meant that existing facilities within the MIL set could be removed. The resulting DTD fragment appears in Appendix 2, together with details of the main changes.

5. TRANSLATION DIFFICULTIES

Problems with the available translator have already been mentioned (inability to handle token lists satisfactorily). In addition, there were other language features missing which made the implementation of the translator a very error-prone exercise. Chief among these were the lack of brackets and any form of loop construct. The first meant that, when wishing to choose between alternative sequences of commands, it was necessary to include the conditional statement with every single line. The second meant that the code could not cope with simply identifying the number of columns in the table from the "cols" attribute. Instead, an assumption had to be made as to the maximum number of columns which would

be handled, and all code which required repetition for each column had to be written out explicitly for each. At the end of each section, the current column was tested against the maximum, and a flag set when the maximum was reached. The remaining code sections could be skipped by preceding each line with a test of this flag.

The resulting translation code is therefore long and almost unreadable (the way in which the LINK section was extended to provide translation facilities meant that the code cannot be commented). It is therefore not included in this paper. If this activity is to be repeated, it will be advisable to design a meta-language with the necessary constructs and generate the translation code by program.

6. SUMMARY

This paper has described the process of document analysis for a specific purpose, and set out the solution adopted. It has proved necessary in practice to accept several compromises, which have detracted from the ideal goal of presenting "pure" semantics. The variety of different ways of marking tables as exhibited by the examples, emphasises the fact that there is no consensus yet on solving the general problem. All that can be said in defence of the solution chosen is that it works.

Finally, it is worth commenting on the use of short references in tables. SGML has a feature whereby a DTD can define selected non-alphanumeric characters to represent tags. The Addison-Wesley DTD defines "@" to represent the start of a row, and "|" to separate the data cells within the row. Other schemes have used the TAB character as cell separator, thus giving the table some visual structure even when tagged. However, this feature only works if there is no requirement to specify attributes on the element. The examples show that using the MIL set as a basis leads to frequent use of attributes, and a table which has a mixture of short references and full elements can be difficult to decipher. It was therefore decided not to include short references in the DTD.

APPENDIX 1

Other DTD Examples

This is a brief description of various published DTDs known to the author which have included markup for tables. Where appropriate, example markup for the table below is included. The descriptions are not meant to be comprehensive. For example, not all the attributes and elements are described. The intention is to give enough information to provide an idea of the underlying table model, sufficient to make the markup intelligible. Also, the author of this paper may have misinterpreted some of the features (it was not possible to gain access to comprehensive manuals in all cases).

The selected table is one of those used in the "Analysis of Tables" chapter.

H1 H1.1	Head1 H2.1	 H3.1
Entry1 Entry2 Entry3	EntryA EntryB EntryC	EntryX EntryY EntryZ

It was chosen to exhibit sufficient complexity for adequate comparison. Examples are simplified by the omission of end tags where possible.

1. Air Transport Association (ATA)

The ATA have published a DTD which adopts the MIL-M-28001A standard for tables. These will be described later.

2. Association of American Publishers (AAP)

The AAP has produced a DTD for Electronic Manuscript Preparation. This DTD distinguishes Simple Tables from Complex Tables, and has a completely different markup set for each (even the initial element is different). Simple Tables (tbl) have a number and title (which span the whole table), followed by a table body (tby) which includes both column headings (th) and data (row), with an optional Source (src) entry at the bottom of the table spanning the whole width (designed for author information). The first element in each row can be designated (by separate tag) as a stub element (which means that it is a label for the row rather than genuinely part of the row data). The number of columns is inferred from the number of column header tags appearing, and there is no method of allowing these headings to span more than one column. Column and row separators are not specified explicitly.

The complex table (ctbl) provides much greater flexibility. The table structure is: Table Head, Table Body, Table Foot. Attributes are used to set the styles for column and row separators, and also alignment of headers and data (both within a column and within a row). Use is made

of the NMTOKENS attribute type, which accepts a string of tokens separated by spaces. Each token refers to an individual column or row.

The Table Head (cthd) contains the table number and title, but also includes the Source information. Column headings and subheadings can be placed here. The defaults for separators and alignments can be changed. The Table Body (ctby) contains rows which either have subheadings or data elements (cte). Four different styles of stub are available for the first cell in a row. Rows of headings can appear between rows of data. Alignment can be varied by row or by individual data cell (using attributes). Headings and data can span columns (start and end column numbers are supplied as separate attributes) and also rows. Complex data structures can occur in a cell, and tables may be nested. The flexibility of the above means that the same presentation can be achieved by a variety of different markup schemes (for example, the column headings could be placed within the cthd or ctby elements).

The example table could be tagged as follows:

```
<ctbl rs="0 s 1 s 2 s F s" cs="0 s 1 s 2 s F s"
      ca="1 l 2 c 3 c">
<cthd>
<cthr><cth rb=1 re=2>H1 H1.1
      <cth rb=1 cb=2 ce=3>Head2
<cthr><ctsh cb=2>H3.1
      <ctsh>H3.2
<ctby>
<ctr><cte>Entry1<cte>EntryA<cte>EntryX
<ctr><cte>Entry2<cte>EntryB<cte>EntryY
<ctr><cte>Entry3<cte>EntryC<cte>EntryZ
</ctbl>
```

3. Addison-Wesley

In his book "SGML: An Author's Guide to the Standard Generalized Markup Language", published by Addison-Wesley, Martin Bryan describes the table markup used for the structures which appear in the book. The number of columns in a table is specified as an attribute on the table element. Within the table, there is the ability to provide a table number (nt), a title (ht), column headings (hc), a table body (bt) and a table foot (ft). The title appears outside the normal table structure (cf caption).

The hc elements can refer to one or more columns (using a "cols" attribute to give the number of columns spanned). Each subsequent hc element will cover one or more columns beside the set specified so far (rather than identifying separate rows). Within the element, each heading is tagged as a cell item (c), which can take a "straddle" attribute specifying how many columns the cell will span (which must fit within the defined hc span). It is also possible to define rows within cells (r) to provide some substructure. The actual presentation of the heading is left to the formatter, so there is no means of specifying alignments, nor row or column separators. In the book, horizontal separators appear just before the bt element and also whenever the r element appears in an hc section. Vertical separators are inserted at each use

of the hc element. Text is left justified unless it spans more than one column (when it is centred in the span). The spanning information is relative to the current position, rather than specified by absolute column number.

In the body of the table, the same row and column elements are used to tag the data (the SGML structure ensures they will be interpreted correctly).

The lack of alignment attribute means that it is only possible to produce a "close match".

```
<table cols=3>
<hc><c>H1 H1.1
<hc cols=2><c straddle=2>Head2
          <r><c>H3.1<c>H3.2</hc>
<tb>
<r><c>Entry1<c>EntryA<c>EntryX
<r><c>Entry2<c>EntryB<c>EntryY
<r><c>Entry3<c>EntryC<c>EntryZ
</table>
```

4. DocBook

This DTD has been written by HaL Computer Systems Inc and O'Reilly and Associates, Inc. The basic structure is: Title (optional), TableSpec, TableColHead, TableRow. Details of the number of columns appears in the content of the TableSpec, rather than as an attribute. TableColHead elements define the (possibly multiline) column headings, and TableRow elements define the each data row. Both consist of TableCell elements which identify the actual information. The only attribute used is an identifier for the whole table. No spanned headings appear (apart from the use of the title element to give a single line full span), and there is no reference to column or row separators. Tables may not be nested.

5. Exoterica Complex Tables

Exoterica have produced a set of elements for table markup, called the Complex Table set (ECT). A table (tbl) consists of an optional title (t.ti) followed by a header section (t.head) and a data section (t.body). Notes and Source information can follow the table. The title is intended to be displayed outside the main table structure (cf caption).

The number of columns in the table is given by a mandatory attribute "cw" on the tbl element. This uses the NUMBERS attribute type, with a number for each column. This number is used as the relative column width, as well as indicating the number of entries. Further attributes define the overall settings for horizontal and vertical separators, and alignments. The information is supplied as a set of tokens using the NMTOKENS attribute type. The tokens are read in pairs: the first item in the pair either indicates a specific row or column by number, or by feature (eg separator between head and body, right hand side of surrounding box, default settings for all rows/columns not explicitly mentioned). A surrounding box can be specified by the individual separator components (top, bottom, sides) or by a "box" attribute. Other attributes specify format information such as overall table width.

Both t.head and t.body have the same substructure, consisting of a set of rows (t.row) with individual items in each cell (t.col). Separators and alignment can be set for each part, column separators can be set for each row, and individual row separators specified for each row. Individual alignment can be set for each cell.

A cell can span more than one column. The "span" attribute gives the number of columns spanned. The span will start from the current column position, so it may be necessary to insert empty t.col elements to reach the correct starting place.

It is possible to indicate a row heading level (4 levels), which identifies the first cell as a "stub". In addition, rows can be flagged as "organisational", either as the first row of a group, or as a summary row (the last row of a group).

Tables may be nested within a cell.

The example table could be tagged as follows:

```
<tbl cw="1 1 1" rs="0 b H b A k F b"
                cs="A b" ha='1 L A c" >
<t.head>
<t.row><t.col>H1  H1.1
       <t.col span=2>
         <tbl cw="1 1" rs="A b" cs="A b" ha="A c">
         <t.body><t.row><t.col span=2> Head1
                 <t.row><t.col>H2.1
                        <t.col>H3.1
         </t.body>
         </tbl>
<t.body>
<t.row><t.col>Entry1<t.col>EntryA<t.col>EntryX
<t.row><t.col>Entry2<t.col>EntryB<t.col>EntryY
<t.row><t.col>Entry3<t.col>EntryC<t.col>EntryZ
</t.body>
</tbl>
```

6. ISO/IEC TR 9573-11

This DTD is part of the ISO Central Secretariat SGML Application, written by Anders Berglund, and has been produced by ISO for use in the production of most of the International Standards Documents, including working drafts and technical reports. It includes features which have been added to permit other standardization bodies to make use of it.

The table (tab) consists of an optional title and description, followed by the main part (tabmat) and possibly following paragraphs. Attributes include layout details such as alignment within column or page and placement relative to the surrounding text. It is also possible to define the table type (eg chart or form). The number of columns required is not requested explicitly, but has to be inferred from the information provided.

Tabmat consist of three parts: tabhead, tabfoot and tabbody. The first two are optional. Attributes define the border and default settings for other attributes used in sub-elements. The border is defined by a string of between one and four characters, separated by spaces, each of which identifies one of the four sides. A complete frame is therefore "T B L R". Similar values are used to define borders for individual rows and cells. Thus the same visual layout can be obtained from many

different settings. The width of a column can be defined by a character string giving a template for each column.

Only one row (arow) is allowed in the heading, whereas the body can have many rows. However, subheadings (and complex substructures within the table) are handled by three attributes: "gridx", "gridy" and "arrange". The method used is complicated to describe, but (briefly) splits a row into subcells, whose relative number and size are specified with "gridx" and "gridy". "arrange" then indicates the manner in which the subcells are grouped. Spans can be handled by this mechanism (see example).

Horizontal and vertical alignment within cells can be specified (again by a CDATA character string). Four header levels are available for tabhead. Clles can be given a domain attribute to indicate such things as subtitles and row headers.

The example table could be tagged as follows:

```
<tab align=center type=table>
<tabmat trules="T B L R"
        crules="R"
        caligns="L C C" >
<tabhead>
<arow gridx="* * *" gridy="* *" arrange="1 2 2 / 3 4 5"
      caligns="L C / L C C" >
<c>H1<c crules="B">Head1<c>H1.1<c>H2.1<c>H3.1
<tabbody>
<arow><c>Entry1<c>EntryA<c>EntryX
<arow><c>Entry2<c>EntryB<c>EntryY
<arow><c>Entry3<c>EntryC<c>EntryZ
</tab>
```

7. MIL-M-28001A

The MIL set of DTDs was developed for the US Department of Defence Computer-aided Acquisition and Logistic Support (CALS) initiative. The original MIL-M-28001 set had very rudimentary table facilities, with no spanning capability. The revised set is able to handle much more complex structures.

The table (table) consists of an optional title (title) and a set of table groups (tgroup). The tgroup element has an attribute giving the number of columns in that particular group, whose substructure consists of a set of column specifiers (colspec), a specification of any spans required (spanspec), a head (thead), foot (tfoot) and body (tbody). Colspec elements may

appear in the head to change settings such as heading alignment. Apart from that, both head and body consist of a series of rows (row) with individual cells (entry). One level of table nesting is provided by replacing the entry element with entrytbl, which has the same structure as the table element, but without permitting another entrytbl element to appear.

Attributes on the table element indicate whether a surrounding frame is required, and whether horizontal and vertical separators will be inserted by default. In addition, there is a "tabstyle" attribute which is intended to refer to a set of table styles defined for CALS. These styles convey further format details.

The colspec element defines the column alignment, the vertical separator to the right of the numbered column and the default row separators within the column. It can also associate a name with the column, which is used in specifying the span. Spanspec refers to the start and end column by name rather than number, and gives a name to the span. This name can be referenced by the entry element. Separators and alignment can be defined for each span. The horizontal separator below a particular row, and the vertical separator to the right of each cell, can be set individually with attributes on the row and entry elements respectively. Each entry can have individual alignments (both vertical and horizontal).

The example table could be tagged as follows:

```
<table frame="all">
<tgroup cols="3" colsep="1">
<colspec colnum="2" colname="c2" align="center">
<colspec colnum="3" colname="c3" align="center">
<spanspec namest="c2" nameend="c3" align="center"
              spanname="c2-3">
<thead>
<row><entry>H1
     <entry rowsep="1" spanname="c2-3">Head1
<row rowsep="1"><entry>H1.1
                <entry>H2.1
                <entry>H3.1
<tbody>
<row><entry>Entry1<entry>EntryA<entry>EntryX
<row><entry>Entry2<entry>EntryB<entry>EntryY
<row><entry>Entry3<entry>EntryC<entry>EntryZ
</table>
```

8. SoftQuad

SoftQuad sell SGML editors, which are capable of working with user supplied DTDs. The latest version includes the ability to handle tables. In order to cope with mapping user-supplied DTDs, SoftQuad have designed a Canonical Form, and all user DTDs are mapped into this. The intention is to be able to provide a visual presentation of the layout, so the Form concentrates on syntax rather than semantics.

There is no single table element. Instead, there are three separate elements: Table Head

(TblHead), Table Body (TblBody) and Table Foot (TblFoot). These can be used jointly or separately in any combination to create the desired form. Each element has exactly the same internal structure, so headings and data are only distinguished by convention (headings would be expected in the TblHead section for example). Attributes on these top level headings specify the physical size which the section will occupy on the page.

Within one of these elements, there are two sections: column definitions (TblCDfs) and rows (TblRows). Within TblCDfs, there is one TblCDef element for each column. Attributes specify the default settings for alignments and vertical separators, with a specific attribute (TopSep) to define the top edge of the surrounding box. These same attributes can be set separately for each column. Similarly, within TblRows there will be one TblRow element for each row, with the individual data cells tagged by TblCell elements. Default row alignments and horizontal separators can be set, with a special tag for the left-hand side of the surrounding box. These settings can be overridden for each row and for each cell. In addition, cell data can span several rows or columns (absolute start and end points specified as attributes).

Once again, the same presentation can be generated in many ways.

```
<TblHead>
<TblCDfs ColSep="VSingle" TopSep="HSingle">
<TblCDef>
<TblCDef HAlign="Center">
<TblCDef HAlign="Center">
<TblRows RowSep="HSingle" LeftSep="VSingle">
<TblRow><TblCell RowSep="HNone">H1
        <TblCell ColStart=2 ColSpan=2>Head1
<TblRow><TblCell>H1.1<TblCell>H2.1<TblCell>H3.1
<TblBody>
<TblCDfs ColSep="VSingle">
<TblCDef>
<TblCDef HAlign="Center">
<TblCDef HAlign="Center">
<TblRows LeftSep="VSingle">
<TblRow><TblCell>Entry1<TblCell>EntryA<TblCell>EntryX
<TblRow><TblCell>Entry2<TblCell>EntryB<TblCell>EntryY
<TblRow RowSep="HSingle">
        <TblCell>Entry3<TblCell>EntryC<TblCell>EntryZ
</TblBody>
```

9. The Text Encoding Intiative (TEI)

TEI provides two elements (table and ext.table) with no further substructure. Instead, it is assumed that a NOTATION statement will provide the information on the data content. This is equivalent to the way in which tables were handled in the first version of the slide set DTD (and is more correct).

APPENDIX 2

Fragment of DTD defining table element.

```
<! --  Based on the MIL-M-28001A table specification >

<!ELEMENT table - O (tgroup) -(table) >

<!ENTITY % yesorno "NUMBER"
        -- setting from body of MIL-M-28001A -->

<!ATTLIST table   tabstyle  NMTOKEN     #IMPLIED
                  frame     (all|none)  "none" >

<!ELEMENT  tgroup - O  (colspec*,spanspec*,thead?,tbody) >

<!ATTLIST  tgroup cols      NUMBER                #REQUIRED
                  colsep    %yesorno;             "0"
                  rowsep    %yesorno;             "0"
                  align     (left|centre|number)  "left" >

<!ELEMENT   colspec     - O      EMPTY  >

<!ATTLIST colspec colnum    NUMBER                #IMPLIED
                  colname   NMTOKEN               #IMPLIED
                  align     (left|centre|number)  #IMPLIED
                  colwidth  CDATA                 #IMPLIED
                  colsep    %yesorno;             #IMPLIED
                  rowsep    %yesorno;             #IMPLIED >

<!ELEMENT   spanspec     - O      EMPTY  >

<!ATTLIST spanspec namest   NMTOKEN               #REQUIRED
                   nameend  NMTOKEN               #REQUIRED
                   align    (left|centre|number)  #IMPLIED
                   colsep   %yesorno;             #IMPLIED
                   rowsep   %yesorno;             #IMPLIED
                   spanname NMTOKEN               #REQUIRED >

<!ELEMENT  thead         - O    (colspec*,row+)   >

<!ELEMENT  tbody         O O    (row+)      >

<!ELEMENT  row           O O    (entry+)     >

<!ATTLIST   row   rowsep    %yesorno;            #IMPLIED >

<!ELEMENT   entry        O O    (#PCDATA)     >
```

```
<!ATTLIST   entry spanname  NMTOKEN                #IMPLIED
                  colsep    %yesorno;              #IMPLIED
                  rowsep    %yesorno;              #IMPLIED
                  align     (left|centre|number) #IMPLIED >


<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -- >


<!-- short references for table. -->

<!ENTITY strow      STARTTAG        "row" >

<!ENTITY endrow     ENDTAG          "row" >

<!ENTITY sthead     STARTTAG        "thead" >

<!ENTITY stbody     STARTTAG        "tbody" >

<!ENTITY stentry    STARTTAG        "entry" >

<!SHORTREF tabmap   ":"       sthead
                    "#"       stbody >

<!USEMAP   tabmap          table   >

<!SHORTREF tabhmap  "{"       strow
                    "}"       endrow
                    "#"       stbody
                    "&#TAB;"  stentry >

<!USEMAP   tabhmap         thead   >

<!SHORTREF tabbmap  "{"       strow
                    "}"       endrow
                    "&#TAB;"  stentry >

<!USEMAP   tabbmap         tbody   >


<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -- >


<!-- Public Entity Insertions for access to special characters
     used in the tables:  "less than or equal", "greater than
     or equal", "unfilled square", "filled bullet".       -- >
```

```
<!ENTITY % ISOtech PUBLIC
   "ISO 8879-1986//ENTITIES General Technical//EN">

<!ENTITY % ISOpub PUBLIC
   "ISO 8879-1986//ENTITIES Publishing//EN">

%ISOtech;

%ISOpub;
```

The main changes to MIL-M-28001A are listed below.

a) The TITLE and SHORTITLE elements are omitted.

b) Where possible, element start and end tags are made optional (the original has all start tags mandatory).

c) Only one TGROUP element is allowed. In fact, this element could be considered redundant, and the attributes merged with those of table, but, in the interests of compatibility, the element is retained.

d) The "tabstyle" and "frame" attributes for table are retained. As stated earlier, local values were defined for "tabstyle" to specify centring and large characters. The single token type (NMTOKEN) was retained for "tabstyle". The values of the "frame" attribute are limited to "all" or "none".

e) Table footers are not required. The TFOOT element is therefore omitted.

f) The "cols", "colsep", "rowsep" and "align" attributes are retained for TGROUP.

g) The values of the "align" attribute are limited to "left", "centre" or "number". The last two settings are additions to the base set. The US spelling "center" is removed.

h) The "char", "charoff", "valign" and security related attributes are removed from all elements.

i) No embedded tables are required, so the ENTRYTBL element is omitted.

j) Each ENTRY element contains no further substructure.

k) The "morerows" and "rotate" attributes are omitted from the ENTRY element.

APPENDIX 3

Examples of Tagged Tables
(taken from chapter 3)

Simple Table

```
<table tabstyle="centre" >
<tgroup cols="3" >
<tbody>
<row>Entry1<entry>EntryA<entry>EntryX
<row>Entry2<entry>EntryB<entry>EntryY
<row>Entry3<entry>EntryC<entry>EntryZ
</table>
```

Single Line Column Heading

```
<table tabstyle="centre" frame="all" >
<tgroup cols="2"  colsep="1">
<colspec colnum="1" align="centre" >
<colspec colnum="2" align="centre" >
<thead>
<row rowsep="1">H1<entry>H2
<tbody>
<row>Entry1<entry>EntryA
<row>Entry2<entry>EntryB
<row>Entry3<entry>EntryC
</table>
```

Multiline Column Heading

```
<table tabstyle="centre" frame="all" >
<tgroup cols="2"  colsep="1">
<colspec colnum="1" align="centre" >
<colspec colnum="2" align="centre" >
<thead>
<row>H1<entry>H2
<row rowsep="1">H1.1<entry>H2.1
<tbody>
<row>Entry1<entry>EntryA
<row>Entry2<entry>EntryB
<row>Entry3<entry>EntryC
</table>
```

Spanned Heading

```
<table tabstyle="centre" >
<tgroup cols="2" >
<colspec colnum="1" colname="c1" >
<colspec colnum="2" colname="c2" >
<spanspec namest="c1" nameend="c2" rowsep="1" align="centre"
          spanname="c1-2" >
<thead>
<row><entry spanname="c1-2">Head1
<tbody>
<row>Entry1<entry>EntryA
<row>Entry2<entry>EntryB
<row>Entry3<entry>EntryC
</table>
```

Full Spanned Heading and Subheading

```
<table tabstyle="centre" frame="all" >
<tgroup cols="3" rowsep="1" colsep="1">
<colspec colnum="1" colname="c1" >
<colspec colnum="2" align="centre" >
<colspec colnum="3" colname="c3" align="centre" >
<spanspec namest="c1" nameend="c3" align="centre"
           spanname="c1-3" >
<thead>
<row><entry spanname="c1-3">Head1
<row>H1.1<entry>H2.1<entry>H3.1
<tbody>
<row>Entry1<entry>EntryA<entry>EntryX
<row>Entry2<entry>EntryB<entry>EntryY
<row>Entry3<entry>EntryC<entry>EntryZ
</table>
```

Partial Spanned Heading and Subheading

```
<table tabstyle="centre" frame="all" >
<tgroup cols="3" colsep="1">
<colspec colnum="2" colname="c2" align="centre" >
<colspec colnum="3" colname="c3" align="centre" >
<spanspec namest="c2" nameend="c3" align="centre"
           spanname="c2-3" >
<thead>
<row>H1<entry rowsep="1" spanname="c2-3">Head1
<row rowsep="1">H1.1<entry>H2.1<entry>H3.1
```

```
<tbody>
<row>Entry1<entry>EntryA<entry>EntryX
<row>Entry2<entry>EntryB<entry>EntryY
<row>Entry3<entry>EntryC<entry>EntryZ
</table>
```

Complex

```
<table>
<tgroup cols="3">
<tbody>
<row>Entry1<entry>EntryA<entry>EntryX
<row><entry><entry>EntryB<entry>EntryY
<row rowsep="1" ><entry><entry>EntryC<entry>EntryZ
<row>Entry2<entry>EntryA<entry>EntryX
<row><entry><entry>EntryB<entry>EntryY
<row rowsep="1" ><entry><entry>EntryC<entry>EntryZ
<row>Entry3<entry>EntryA<entry>EntryX
<row><entry><entry>EntryB<entry>EntryY
<row><entry><entry>EntryC<entry>EntryZ
</table>
```