Accessible tables in docBook 5.1 using the expanded CALS table model

Nathalie Sequeira

Abstract

Usage guide to the expanded CALS table model in DocBook 5.1, with notes for documentation changes to enhance its usability.

Table of Contents

Why do this?

The increasingly widespread usage of texts in their digital form has hugely positive implications for the inclusion of people with disabilities. Access to information is becoming much easier, especially for blind people, since digital texts can be read directly by a screen reader without any extra manipulation (e.g. conversion to Braille, scannning textbooks so they can be read aloud by corresponding softwares, etc.).

However, to guarantee that the blind user's experience is as smoothe as possible, certain helpers must be in place to facilitate navigation and the correct interpretation of a text by screen readers. Structural markup of headings, lists and tables, for example, are central to this. Thus, docBook innately already has much of what is needed in place to create texts that are universally accessible.

In the specific case of tables, it is vital to attribute table headers to the table data. Where this is usually achieved visually by distinct header styles, someone who cannot see these cues will need programmatic indications of which cells contain headers, and how they relate to the data presented in the table. Without these semantic markers, screen readers will read table data linearly, that is, from left to right, row by row, thus making it very hard to understand the data and its interrelations.

DocBook 5.1 and its official Publishers variant set out to offer this needed markup by expanding the CALS table model in such a way that these semantic relationships can be expressed and transferred to other formats the docBook is converted to, very similarly to how the HTML table model already can achieve this.

The following walk-through shows the new possibilities of the CALS table model in use.

Walk-through by example

In marking up a CALS table for screen reader accessibility, there are two steps to be taken, depending on the table structure's complexity:

1. identify the table's headers

2. explicitly associate table data with its headers

Identify the table's headers

In very simple tables with only one row or column of headers, this mechanism alone is sufficient to allow screen readers to accociate data with its respective headers.

Identify column headers

To identify column headers, the relevant rows are wrapped in the `thead` element.

Table 1. example: a simple table with column headers

Mark's points	Peter's points	Cindy's points
11,123.45	11,012.34	10,987.64

```
<table frame="all">
    <title>example: a simple table with column headers</title>
    <tgroup cols="3">
        <thead>
            <row>
                <entry>Mark's points</entry>
                <entry>Peter's points</entry>
                <entry>Cindy's points</entry>
            </row>
        </thead>
        <tbody>
            <row>
                <entry>11,123.45</entry>
                <entry>11,012.34</entry>
                <entry>10,987.64</entry>
            </row>
        </tbody>
    </tgroup>
</table>
```

Without any markup of the table headers, a sreen reader may just read the table's contents as:

> Mark's points Peter's points Cindy's points 11,123.45 11,012.34 10,987.64.

With the headers marked up, screen readers will attribute the headers like this:

> Mark's points 11,123.45, Peter's points 11,012.34, Cindy's points 10,987.64.

Clearly, the second variant is much more easily understood, and thus well worth the little effort required to appropriately markup the table in the docBook source.

Identify row headers

DocBook now offers two alternatives to identify row headers in CALS tables:

- if only the first column contains row headers, set the `rowheader` attribute to the `firstcol` value in the `table` element.

Table 2. example: a simple table with first row headers

Mark's points	11,123.45
Peter's points	11,012.34
Cindy's points	10,987.64

```
<table frame="all" rowheader="firstcol">
    <title>example: a simple table with first row headers</title>
    <tgroup cols="2">
        <tbody>
            <row>
                <entry>Mark's points</entry>
                <entry>11,123.45</entry>
            </row>
            <row>
                <entry>Peter's points</entry>
                <entry>11,012.34</entry>
            </row>
            <row>
                <entry>Cindy's points</entry>
                <entry>10,987.64</entry>
            </row>
        </tbody>
    </tgroup>
    <caption><para>Captions are allowed!</para></caption>
</table>
```

- if more than one column contains row headers, set the `rowheader` attribute on the relevant `colspec` declarations to `headers`.

Table 3. example: a table with two columns of row headers

points	Mark	11,123.45
	Peter	11,012.34
	Cindy	10,987.64

```
<table frame="all">
    <title>example: a table with two columns of row headers</title>
    <tgroup cols="3">
        <colspec colname="c1" colwidth="1.0*" rowheader="headers"/>
        <colspec colname="c2" colwidth="1.0*" rowheader="headers"/>
        <colspec colname="c3" colwidth="1.0*"/>
        <tbody>
            <row>
```

```
                <entry morerows="2">points</entry>
                <entry>Mark</entry>
                <entry>11,123.45</entry>
            </row>
            <row>
                <entry>Peter</entry>
                <entry>11,012.34</entry>
            </row>
            <row>
                <entry>Cindy</entry>
                <entry>10,987.64</entry>
            </row>
        </tbody>
    </tgroup>
</table>
```

Note: the examples here are mere constructions to demonstrate docBook functionaliy options. In the wild, the first variant would be preferred to the second from an accessibility standpoint. Also, in the second example, the spanned entry "points" would have to include a `scope="rowgroup"` to make its attribution unmistakably clear (left out here for clarity's sake).

Explicity associate table data with its headers

This step is necessary for more complex tables, for example containing row AND column headers, multiple column header rows, or spanned entries.

Analagously to the HTML model, DocBook CALS tables now offer two mechanisms to achieve this:

- `scope`

  The scope attribute is used on table entries functioning as headers to explicitly define the range of entries they apply to (a row, a column, a row group, or a column group).

  This method is best suited to tables of medium complextiy.

Table 4. example: assuring complex table accessibility using `scope`

	points	
	expected	**actual**
Mark	10,000	11,123.45
Peter	9,000	11,012.34
Cindy	10,000	10,987.64

```
<table frame="all" rowheader="firstcol">
<title>
    example: assuring complex table accessibility using <code>scope</code>
</title>
<tgroup cols="3">
    <colspec colname="c1" colwidth="1.0*"/>
    <colspec colname="c2" colwidth="1.0*"/>
```

```
        <colspec colname="c3" colwidth="1.0*"/>
        <thead>
            <row>
                <entry morerows="1"></entry>
                <entry namest="c2" nameend="c3" scope="colgroup">points</entry>
            </row>
            <row>
                <entry scope="col">expected</entry>
                <entry scope="col">actual</entry>
            </row>
        </thead>
        <tbody>
            <row>
                <entry scope="row">Mark</entry>
                <entry>10,000</entry>
                <entry>11,123.45</entry>
            </row>
            <row>
                <entry scope="row">Peter</entry>
                <entry>9,000</entry>
                <entry>11,012.34</entry>
            </row>
            <row>
                <entry scope="row">Cindy</entry>
                <entry>10,000</entry>
                <entry>10,987.64</entry>
            </row>
        </tbody>
    </tgroup>
    </table>
```

As a bareboned table, it would be read by a screen reader like so:

> points expected actual. Mark 10,000 11,123.45. Peter 9,000 11,012.34. Cindy 10,000 10,987.64.

which really does not make much sense at all. But by using the markup as shown above, a screen reader will be able to see order in the chaos and will read:

> Mark points expected 10,000 Mark points actual 11,123.45. Peter points expected 9,000 Peter points actual 11,012.34. Cindy points expected 10,000 Cindy points actual 10,987.64.

While this leans slightly redundantly towards the verbose, it is also by far more clear, allowing for quick understanding even if the reader cannot see the table's spatial layout.

- `xml:id` and `headers`

  With this dynamic duo, a very granular association of table data to its headers can be accomplished. The header entries are identified via `xml:id` attributes, which are then referenced in the `headers` attribute on data entries.

  This method is best suited for very complex tables that cannot be otherwise simplified.

Table 5. example: assuring complex table accessibility using `headers`

	points	
	expected	**actual**
Mark	10,000	11,123.45
Peter	9,000	11,012.34
Cindy	10,000	10,987.64

```
<table frame="all" rowheader="firstcol">
<title>
    example: assuring complex table accessibility using <code>headers</code>
</title>
<tgroup cols="3">
    <colspec colname="c1" colwidth="1.0*"/>
    <colspec colname="c2" colwidth="1.0*"/>
    <colspec colname="c3" colwidth="1.0*"/>
    <thead>
        <row>
            <entry morerows="1"></entry>
            <entry namest="c2" nameend="c3" xml:id="pts">points</entry>
        </row>
        <row>
            <entry xml:id="exp" headers="pts">expected</entry>
            <entry xml:id="act" headers="pts">actual</entry>
        </row>
    </thead>
    <tbody>
        <row>
            <entry xml:id="name1">Mark</entry>
            <entry headers="name1 exp pts">10,000</entry>
            <entry headers="name1 act pts">11,123.45</entry>
        </row>
        <row>
            <entry xml:id="name2">Peter</entry>
            <entry headers="name2 exp pts">9,000</entry>
            <entry headers="name2 act pts">11,012.34</entry>
        </row>
        <row>
            <entry xml:id="name3">Cindy</entry>
            <entry headers="name3 exp pts">10,000</entry>
            <entry headers="name3 act pts">10,987.64</entry>
        </row>
    </tbody>
</tgroup>
</table>
```

To do: Documentation

To make the documentation clear concerning the different attribute values for rowheader depending on context, I suggest the following wording:

docBook Schema / docBook Publishers Schema

db.rowheader.attribute =

```
 ## Indicates whether or not the entries in columns should be considered row headers
 attribute rowheader {

   ## Indicates that entries in the first column of the table are functionally
      row headers (analogous to the way that a thead provides column headers).
      Applies when rowheader is used as a table attribute.
   "firstcol"
   |
    ## Indicates that entries of a column described via colspec are functionally
       row headers (for cases with more than one column of row headers).
       Applies when rowheader is used as a colspec attribute.
    "headers"
   |
    ## Indicates that entries in columns have no special significance
       with respect to column headers.
    "norowheader"
 }
```

The Definitive Guide

Being as it is the definitive guide, it may be useful to include a usage guide along the lines of the one above. However, I could not find "the perfect spot" for this.

The same changes are to be made in the appropriate spots of the DocBook Publishers Definitive Guide.

db.cals.table (http://www.docbook.org/tdg51/en/html/cals.table.html []), and db.cals.informaltable (http://www.docbook.org/tdg51/en/html/cals.informaltable.html)

rowheader — Indicates whether or not the entries in the first column should be considered row headers

Enumerated values	Explanation
“firstcol”	Indicates that entries in the first column of the table are functionally row headers (analogous to the way that a thead provides column headers).

Enumerated values	Explanation
“headers”	Do not use in the `table` context. (For use in the context of `colspec` only.)
“norowheader”	Indicates that entries in the first column have no special significance with respect to column headers.

colspec (http://www.docbook.org/tdg51/en/html/colspec.html)

rowheader

Indicates whether or not the entries in the specified column should be considered row headers

Enumerated values	Explanation
“firstcol”	Do not use in the `colspec` context. (For use in the context of `table` only.)
“headers”	Indicates that entries in the specified column are functionally row headers (analogous to the way that a thead provides column headers).
“norowheader”	Indicates that entries in the column have no special significance with respect to column headers.

To do: XSL Transformations

The following transformations are to be added:

- HTML
  - if the colspec for an entry's column specifies rowheaders, `entry` --> `th`
  - assure that empty entries in row-or column-header groups are transformed to empty `td`'s
  - transfer the `scope` and `headers` attribute into the HTML as is.
- ePub?

  How must tables be marked up in ePub to be screen reader accessible?
- FO?

  How must tables be marked up in PDF to be screen reader accessible?
  - transform <caption> elements
  - transform colspec-defined header columns (analagously to firstcol, as already implemented?)