May 26, 2006

ODF Access Requirements

OASIS ODF Accessibility Subcommittee

Executive Summary

The ODF Accessibility Subcommittee has identified 9 accessibility issues in ODF 1.0, and proposes candidate solutions to them. With these changes, we believe that ODF will meet or exceed the accessibility support provided in all other office file formats as well as that specified in the W3C Web Content Accessibility Guidelines 1.0.

Furthermore, these modifications will enable ODF to support the authoring of DAISY digital talking books, a worldwide standard used by blind, low vision, learning disabled, and other print impaired communities.

The recommended changes address:

- Alternative text for non-text objects (3 recommendations)
- Proper association of captions to captioned content
- Encoding of pagination information
- Preservation of table semantic structure imported from other file formats
- Proper encoding of authored table header content
- Author-defined logical navigation of page objects in presentations
- Provision of alternative text hints for hyperlinks

Furthermore, we request that the appropriate text be added to the ODF specification to indicate how this accessibility meta data is mapped by the authoring tool to a platform accessibility API as well as their accessibility applicability in the specification.

To fully address the needs of people with disabilities in using ODF, an ODF application must meet a number of accessibility requirements as well. ODF application developers should be provided with implementation guidelines to meet these requirements.

1.0 Requirement: Provide for soft page breaks in the specification

Users are unable to share common page numbering when rendered with different applications, on different systems. This negatively impacts conversions to DAISY digital talking books which utilize page-based navigation as a common reference model to printed material, as well as Braille and large print uses. A common use of these alternate formats is in classrooms, where instructors tell students to turn to a specific page in a book.

When a document is paginated the soft page break elements should be exported. We suggest this be implemented by introducing a new XML tag in writer similar to hard page breaks for soft page breaks.

2.0 Requirement: Correct wording in specification to require table header structural markup

Users are unable to recognize all of the table headers that are created as table headers in ODF 1.0.

These two sections require changes in the ODF 1.0 specification as defects were discovered during assistive technology interoperability testing with tables. This will impact office applications today as they have been found to incorrectly use styling vs. declarative markup for indicating table headers. Declarative markup is essential for determining proper table structure semantics. Styling does not indicate semantic intent.

8.2.2 Header Columns

For accessibility purposes, header information is needed. Therefore, any columns designated as headers by the author must be tagged as such by encapsulating them within the <table:table-header-columns>. Using style is insufficient. If a table does not fit on a single page, a set of adjacent table columns can be automatically repeated on every page. To do so, their columns descriptions have to be included in a `<table:table-header-columns>` element. ~~Descriptions of columns that shall not be repeated on every page can be included into a `<table:table-columns>` element, but don't have to.~~ A table must not contain more than one `<table:table-header-columns>` element~~, and a `<table:table-columns>` must not follow another `<table:table-columns>` element. T~~with the only exception ~~are~~of tables that contain grouped columns (see 8.2.3). Such tables contain more than one `<table:table-header-columns>` element, provided that they are contained in different column groups and the columns contained in the elements are adjacent.

Applications that do not support header columns have to process header column descriptions the same way as non header column descriptions.

The `<table:table-header-columns>` and `<table:table-columns>` element are very similar to [HTML4]'s `<THEAD>` and `<TBODY>` elements for rows.

```
<define name="table-table-header-columns">
    <element name="table:table-header-columns">
        <oneOrMore>
            <ref name="table-table-column"/>
        </oneOrMore>
    </element>
</define>

<define name="table-table-columns">
    <element name="table:table-columns">
        <oneOrMore>
            <ref name="table-table-column"/>
        </oneOrMore>
    </element>
</define>
```

8.2.4 Header Rows

For accessibility purposes, header information is needed. Therefore, any rows designated as headers by the author must be tagged as such by encapsulating them within the <table:table-header-rows>. Using style is insufficient. If a table does not fit on a single page, a set of adjacent table rows can be automatically repeated on every page. To do so, their row elements have to be included in a `<table:table-header-rows>` element. ~~Rows that shall not be repeated on every page can be included into a <table:table-rows> element, but don't have to.~~ A table must not contain more than one `<table:table-header-rows>` element~~, and a <table:table-rows> must not follow another <table:table-rows> element~~. The ~~only~~one exception to this is~~are~~ a table~~s~~ that contains grouped rows (see 8.2.5). Such a table~~s~~ contains more than one `<table:table-header-rows>` element, provided that they are contained in different row groups and the rows contained in the elements are adjacent.

Applications that do not support header rows have to process header rows the same way as non header rows.

The `<table:table-header-rows>` and `<table:table-rows>` element are very similar to [HTML4]'s `<THEAD>` and `<TBODY>` elements.

```
<define name="table-table-header-rows">
    <element name="table:table-header-rows">
        <oneOrMore>
            <ref name="table-table-row"/>
        </oneOrMore>
    </element>
</define>

<define name="table-table-rows">
    <element name="table:table-rows">
        <oneOrMore>
            <ref name="table-table-row"/>
        </oneOrMore>
    </element>
</define>
```

3.0 Requirement: Provide for author-specified, logical navigation in presentations

Authors are unable to indicate a logical navigation order for traversing through objects in ODF presentations as distinct from z-order. The inability to specify a logical navigation order makes it difficult for some users to properly understand a drawing or presentation. For example, if a presentation was designed such that groups of objects represented a logical process to be followed, a blind user would not be able to follow the correct sequence unless specified.

We need a mechanism to allow the author to specify a logical, intent-based, navigation order independent of the default document navigation order. We suggest that a nav-order attribute be provided in **<draw:page>** so that the author may specify the navigation order of the presentation slide or drawing. This optional attribute should only be specified once the author chooses to provide a navigation order. The suggested schema changes for the nav-order attribute addition to <draw:page> are defined below. The nav-order targets encompass all drawing elements and embedded objects unless

they are embedded in a **<draw:g>**. All drawing elements and embedded objects must be assigned an XML id and they must appear in the nav-order list. Once a navigation order has been specified by the author, all new drawing objects shall be assigned an XML id and placed in the nav-order list. Our UI suggestion is that user agents, which employ a navigation tool, allow the author to selectively choose one or more objects and alter their position in the navigation order as shown here:

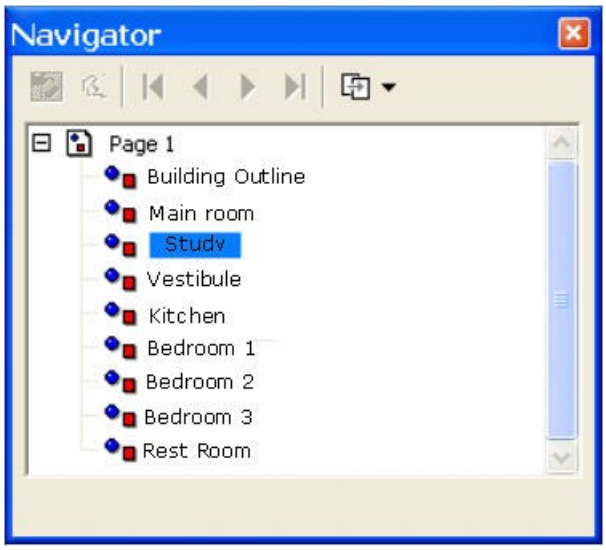

figure 1

The following are the suggested schema modifications:

9.1.4

...

The `draw:id` attribute assigns a unique ID to a drawing page.

```
<define name="draw-page-attlist" combine="interleave">
    <optional>
        <attribute name="draw:id">
            <ref name="ID"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:nav-order">
            <ref name="IDREFS">
        </attribute>
    <optional>
</define>
```

The draw:nav-order attribute defines a logical navigation sequence based on a collection of unique IDREFs. If this optional attribute is present, it must include all graphic elements not contained within a <draw:g> tag. This attribute should reflect the intentional ordering of graphics as set by the document author.

16.1 Data Types

The following data types are used within this specification:

- W3C Schema data types as defined in [xmlschema-2] (referenced by `<ref>` elements named the same as the corresponding data types)
  - string
  - date
  - time
  - dateTime
  - duration
  - integer
  - nonNegativeInteger
  - positiveInteger
  - double
  - anyURI
  - base64Binary
  - ID
  - IDREF
  - IDREFS

Relax-NG definitions for the W3C schema data types:

```
<define name="string">
    <data type="string"/>
</define>
<define name="date">
    <data type="date"/>
</define>
<define name="time">
    <data type="time"/>
</define>
<define name="dateTime">
    <data type="dateTime"/>
</define>
<define name="duration">
    <data type="duration"/>
</define>
<define name="integer">
    <data type="integer"/>
</define>
<define name="nonNegativeInteger">
    <data type="nonNegativeInteger"/>
</define>
<define name="positiveInteger">
    <data type="positiveInteger"/>
</define>
<define name="double">
    <data type="double"/>
</define>
<define name="anyURI">
    <data type="anyURI"/>
</define>
<define name="base64Binary">
    <data type="base64Binary"/>
</define>
<define name="ID">
    <data type="ID"/>
</define>
<define name="IDREF">
    <data type="IDREF"/>
</define>
<define name="IDREFS">
    <data type="IDREFS"/>
</define>
```

4.0 Requirement: Alternative text for non-text image- map elements

Image maps do not provide for alternative text that is essential for accessibility.

We recommend that an **<svg:title>** element be provided as an optional element to the following:

draw:area-rectangle
draw:area-circle
draw:area-polygon

Text should be added to these elements as follows:

<svg:title> is used as a short accessible name. When transcoding from another document format to ODF the short names, like HTML's alt text on the <img> tags shall shall be mapped to the **<svg:title>** element. Alternative text in Microsoft Office is considered a short name and should be mapped accordingly.

<svg:desc> is used for the long description in support of accessibility.

5.0 Requirement: Alternative text for Drawing layer

Drawing layers do not provide for alternative text that is essential for accessibility.

We recommend that **<svg:title>** element be provided as optional element to **<draw:layer>**. This element must apply to all ODF document formats for which **<draw:layer>** is used.

Text should be added as follows:

<svg:title> is used as a short accessible name. When transcoding from another document format to ODF the short names, like HTML's alt text on the <img> tags shall shall be mapped to the **<svg:title>** element. Alternative text in Microsoft Office is considered a short name and should be mapped accordingly.

<svg:desc> is used for the long description in support of accessibility.

6.0 Requirement: Alternative text for drawing objects (line 5926 of spec.)

Drawing objects do not provide for alternative text that is essential for accessibility.

The **<svg:title>** and **<svg:desc>** elements must be provided as optional elements to all drawing shape elements defined below for all ODF document formats for which these drawing shapes are used.

The following are the drawing shape elements:

<draw:rect>
<draw:line>
<draw:polyline>
<draw:polygon>
<draw:regular-polygon>
<draw:path>
<draw:circle>
<draw:ellipse>
<draw:g>
<draw:page-thumbnail>
<draw:frame>
<draw:measure>
<draw:caption>
<draw:connector>
<draw:control>
<dr3d:scene>
<draw-custom-shape>

Text should be added to these elements as follows:

<svg:title> is used as a short accessible name. When transcoding from another document format to ODF the short names, like HTML's alt text on the <img> tags shall shall be mapped to the **<svg:title>** element. Alternative text in Microsoft Office is considered a short name and should be mapped accordingly.

<svg:desc> is used for the long description in support of accessibility.

User agents supporting platform accessibility APIs should follow the following conventions for supporting the accessible name, accessible description (accessible help on Windows systems), and describedBy relationships:

If an **<svg:title>** element is provided it should map to the accessible name. If not, the name should use the text referenced the text element identified by the **draw:describedby** attribute. **<svg:desc>** must be used to support the accessible description. User agents shall not manufacture names for the **<svg:title>** element, such as using the drawing object followed by a cardinal number in a string as it is used for accessibility. Name assignments such as these provide no semantic meaning to the user.

Guidance for authors:
Authors should not assign names to objects having no semantic value. If no name is assigned the caption text will be used in its place. **<svg:title>** shall take precedence over the caption text for accessible name assignment by the user agent.

Assignment of the long description should only be necessary when a drawing object is significantly complex and the user needs more information to describe it. Long descriptions would be more applicable to drawing groupings than basic drawing shapes.

Authoring tool responsibility for presenting and prompting for the **<svg:title>** and **<svg:desc>**:

Authoring tools should provide an option from an objects context menu to allow the user to enter the text for either of these elements as a minimum. More proactive authoring tools should have a facility for prompting the author for this text. Since **<svg:desc>** is a long description, a text area vs. a text field should be used to prompt the user accordingly in GUI-based authoring tools like Workplace and Open Office.

Navigation tools, such as in Workplace and Open Office, used to list the objects in the view should provide
the type of object followed by the contents of **<svg:title>**. The title must have been entered by the author.

For **<draw:g>** elements the drawing objects which are members of the group should visible only when the group is expanded.

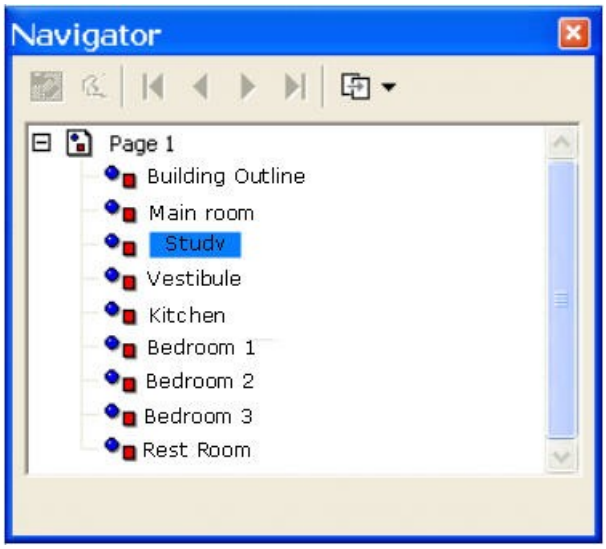

figure 2

7.0 Requirement: Establish clear relationships between objects and their captions

Captions are not clearly associated with the drawing objects which they caption and this is needed for accessibility.

Establish clear relationship between a drawing objects and its caption by including a new optional **draw:describedby** attribute to the following drawing objects.

<draw:rect>
<draw:line>
<draw:polyline>
<draw:polygon>
<draw:regular-polygon>
<draw:path>
<draw:circle>
<draw:ellipse>

<draw:g>
<draw:page-thumbnail>
<draw:measure>
<draw:caption>
<draw:connector>
<draw:control>
<dr3d:scene>
<draw-custom-shape>
<dr3d:scene>
<draw:frame>

draw:describedby shall take a value of IDREF. The value for **draw:describedby** attribute shall be the target id assigned to the **<text:p>** used to represent the corresponding caption. As text:p is an XML element it may have an ID assigned by default. The following attribute list should be included as optional to the above drawing objects:

<define name="common-draw-describedby-attlist" combine="interleave">
<attribute name="draw:desribedby">
<ref name="IDREF"/>
</attribute>
</define>

When a caption is assigned by a user agent, an id must be assigned to the element containing the text used to caption a drawing element. The drawing element being captioned must then be assigned the **draw:describedby** attribute with an IDREF equivalent to the id of the captioning text thus establishing a relationship between the captioned text and the object captioned as needed for accessibility. Removing the caption should result in removing the **draw:describedby** attribute of the object that was being captioned.

If the user agent supports a platform which provides a **draw:describedby** relationship in its accessibility API, this relationship for captions should be used to fulfill the relationship.

8.0 Requirement: Establish text hints for hypertext links

Hypertext links do not provide hints as to the destination of a link. This is required for accessibility so that users may make informed decisions. This also a W3C Web Content Accessibility Guideline requirement.

We recommend that the **<svg:title>** element must be provided as an optional element to **<text:a>**.

<svg:title> is used as a short accessible description for hint text. When transcoding from another document format to ODF the alt text, shall be mapped to the **<svg:title>** element. When exporting ODF documents to HTML, the contents of title text should be mapped to title attribute text on HTML anchor tags. As a minimum, authoring tools should provide a mechanism to provide the hint text.

The title text should be made accessible to the assistive technology and user. The user agent should allow for programmatic access through standard accessibility APIs such as the accessible description. Users should experience visible access to the hint text via the keyboard or mouse.

9.0 Requirement: Provide for structured tables in presentations

Users importing non-ODF slides that contain tables need access to the table structure via their assistive technology. Native table support with access to full semantic structure will be addressed in a future release of the ODF specification. Meanwhile, tables imported into ODF from another file format must have their structure preserved.

We suggest that ODF applications be modified immediately to import tables in presentation as embedded spreadsheets. We further suggest that in the future tables be made a first class object within presentations, similar to they way the are implemented in Writer. Please see the Florian Reuter draft proposal in appendix 1.0 which addresses the accessibility requirements we have identified.

Miscellaneous Corrections to ODF 1.0 Specification

Section 9.2.15 Fix incorrect documentation in z-index

We believe the following may be an oversight when specifying z-index.

Z-Index

Drawing shapes are rendered in a specific order. In general, the shapes are rendered in the order in which they appear in the XML document. To change the order, use the ~~svg:width and svg:height~~draw:z-index attribute.

This attribute is optional.

```
<define name="common-draw-z-index-attlist">
    <optional>
        <attribute name="draw:z-index">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Appendix

1.0 Florian Reuter draft proposal for future table support

Problem:

Currently tables are not supported within presentations. According to the OpenDocument specification tables can only be inserted to presentations by surrounding them with a text box.
What is needed is a table support in OpenDocument, such that tables can be put to presentations directly. One important issue here is to preserve accessibility, i.e. is it possible to navigate through tables with e.g. a screen reader.

Proposed Enhancement

We propose the following enhancements to the OpenDocument specification and the OpenDocument schema:

9.3 Frames

Modify the specification of frames, such that tables can also appear in frames:

```
<define name="draw-frame">
   <element name="draw:frame">
      <ref name="common-draw-shape-with-text-and-styles-attlist"/>
      <ref name="common-draw-position-attlist"/>
      <ref name="common-draw-rel-size-attlist"/>
      <ref name="presentation-shape-attlist"/>
      <ref name="draw-frame-attlist"/>
      <zeroOrMore>
         <choice>
            <ref name="draw-text-box"/>
            <ref name="draw-image"/>
            <ref name="draw-object"/>
            <ref name="draw-object-ole"/>
            <ref name="draw-applet"/>
            <ref name="draw-floating-frame"/>
            <ref name="draw-plugin"/>
            <ref name="table-table"/>
         </choice>
      </zeroOrMore>
      <optional>
         <ref name="office-event-listeners"/>
      </optional>
      <zeroOrMore>
         <ref name="draw-glue-point"/>
      </zeroOrMore>
      <optional>
         <ref name="draw-image-map"/>
      </optional>
      <optional>
         <ref name="svg-desc"/>
      </optional>
      <optional>
         <choice>
            <ref name="draw-contour-polygon"/>
            <ref name="draw-contour-path"/>
         </choice>
      </optional>
   </element>
</define>
```

Interoperability discussion

Presentation applications allow graphic-properties on table cells. This means, that styles referenced by tables cells may contain graphic-properties.

Example

Consider for example the following table:

Header1	Header2	Header2
A1	A2	A3
B1	B2	B3
C1	C2	C3

The above table would be encoded in an OpenDocument spreadsheet as follows:

```
<office:body>
 <office:presentation>
  <draw:page>
   <draw:frame svg:x="5cm" svg:y="7cm">
   <table:table table:name="SampleTable" table:style-name="Table1">
    <table:table-column table:style-name="Table1.Column" table:number-columns-repeated="3"/>
    <table:table-header-rows>
     <table:table-row>
      <table:table-cell table:style-name="Table1.H" office:value-type="string">
       <text:p>Header1</text:p>
      </table:table-cell>
      <table:table-cell table:style-name="Table1.H" office:value-type="string">
       <text:p>Header2</text:p>
      </table:table-cell>
      <table:table-cell table:style-name="Table1.H" office:value-type="string">
       <text:p>Header2</text:p>
      </table:table-cell>
     </table:table-row>
    </table:table-header-rows>
    <table:table-row>
     <table:table-cell table:style-name="Table1.B" office:value-type="string">
      <text:p>A1</text:p>
     </table:table-cell>
     <table:table-cell table:style-name="Table1.B" office:value-type="string">
      <text:p>A2</text:p>
     </table:table-cell>
     <table:table-cell table:style-name="Table1.B" office:value-type="string">
      <text:p>A3</text:p>
     </table:table-cell>
    </table:table-row>
    <table:table-row>
```

```
    <table:table-cell table:style-name="Table1.B" office:value-type="string">
     <text:p>B1</text:p>
    </table:table-cell>
    <table:table-cell table:style-name="Table1.B" office:value-type="string">
     <text:p>B2</text:p>
    </table:table-cell>
    <table:table-cell table:style-name="Table1.B" office:value-type="string">
     <text:p>B3</text:p>
    </table:table-cell>
   </table:table-row>
   <table:table-row>
    <table:table-cell table:style-name="Table1.B" office:value-type="string">
     <text:p>C1</text:p>
    </table:table-cell>
    <table:table-cell table:style-name="Table1.B" office:value-type="string">
     <text:p>C2</text:p>
    </table:table-cell>
    <table:table-cell table:style-name="Table1.B" office:value-type="string">
     <text:p>C3</text:p>
    </table:table-cell>
   </table:table-row>
  </table:table>
  </draw:frame>
 </draw:page>
</office:presentation>
</office:body>
```

with the following styles:

```
<style:style style:name="Table1" style:family="table">
  <style:table-properties style:width="15cm" table:align="margins"/>
</style:style>
<style:style style:name="Table1.Column" style:family="table-column">
  <style:table-column-properties style:column-width="5cm"/>
</style:style>
<style:style style:name="Table1.H" style:family="table-cell">
  <style:table-cell-properties fo:padding="0.097cm" fo:border-left="0.002cm solid #000000"
fo:border-right="none" fo:border-top="0.002cm solid #000000" fo:border-bottom="0.002cm solid
#000000"/>
  <style:graphic-properties draw:fill="gradient" draw:fill-color="#bbe0e3" draw:fill-gradient-
name="Gradient_20_7"/>
</style:style>
<style:style style:name="Table1.B" style:family="table-cell">
  <style:table-cell-properties fo:padding="0.097cm" fo:border="0.002cm solid #000000"/>
</style:style>
```

Purpose of tables

In the current OpenDocument specification tables within presentations are represented based on shapes. This is not acceptable regarding accessibility issues. All OpenDocument processing entities SHOULD use the proposed table enhancement.

The new table feature of OpenDocument SHOULD also be reflected within OpenDocument processing entities such that the accessibility purpose of the proposed feature is preserved.