

# ECMA-376 Office Open XML (DIS 29500)

## Accessibility Guidelines - Draft - January 8, 2008

### Abstract

This document is a guide for applications that support DIS 29500 (ECMA 376 Office Open XML) specification with the goal of encouraging the creation of accessible Office Open XML documents. Office Open XML provides a rich infrastructure for creating content that meets the needs of people with disabilities. This document's guidance must be followed in order to ensure Office Open XML implementations are consistent with respect to their support for accessibility at both the application and output level. Authors and developers are encouraged to follow these guidelines in order to enable users with disabilities to consume content or to extract the full meaning of Office Open XML documents.

### Status and Contributors

This document was authored and reviewed by the following members of the accessibility community:

#### ***Markus Gylling – Technical Development Coordinator DAISY Consortium***

- Swedish Library of Talking Books and Braille
- Board Member, National Information Standards Organization
- Selected Papers – ANSI/NISO Z39.86-2005 “Specifications for the Digital Talking Book”, Markus Gylling, the DAISY Consortium and the Swedish Library of Talking Books and Braille - <http://www.daisy.org/z3986/2005/Z3986>

#### ***George Kerscher – Secretary General DAISY Consortium***

- George chaired the DAISY/NISO Standards Committee.
- US National File Format Technical Panel for Secretary of Education
- 2004 Harry Murphy Catalyst Award Winner
- 2001 Dayton Forman Award
- 1998 US News and World Report Innovator of the Year
- Selected Papers
  - "Implications of Digital Talking Books and Beyond", George Kerscher. National Federation of the Blind presentation 1999. See <http://www.nfb.org/bm00/bm0001/bm000114.htm>
  - "Beyond Gutenberg", Janina Sajka and George Kerscher, 2000, American Foundation for the Blind. See <http://www.afb.org/ebook.html>

**Reed Shaffner – Accessibility Product Manager, Microsoft Office**

- Project Director (with Sonata Software Limited and Daisy Consortium) of the Office Open XML to DAISY Translator at <http://sourceforge.net/projects/openxml-daisy>

**Dana Simberkoff – Vice President, HiSoftware® (Accessibility, Regulatory Compliance)**

- Vice President of HiSoftware®, a leading provider of software and services that test, repair, monitor and enforce audited Web content, quality, and regulatory compliance.
- HiSoftware's accessibility solutions include AccMonitor™, AccRepair, AccVerify, Hi-Caption and compliance software designed to ensure that various technologies comply with accessible technology needs.

**Rob Sinclair – Group Program Manager, Microsoft Accessibility Business Unit**

- Architect, Microsoft User Interface (UI) Automation

**Robert B. Yonaitis, CTO and Founder HiSoftware® (Accessibility, Regulatory Compliance)**

- Yonaitis' role at HiSoftware involves technology direction, product architecture, partner development, and standards development. HiSoftware® is a leading provider of software and services that test, repair, monitor and enforce audited Web content, quality, and regulatory compliance.
- His work led to HiSoftware being chosen by the National Multiple Sclerosis Society and the Engineering Society of Detroit as the 2003 da Vinci Award for Electronics and Information Technology
- Selected Publications:  
*Understanding Accessibility* - 2002, 2007. ISBN - 1-930616-03-1

We welcome additional feedback and comments.

## The Value of Creating an Accessible Office Open XML Implementation

There are numerous benefits associated with creating an accessible Office Open XML implementation. Office Open XML provides a rich infrastructure for creating content that meets the needs by people with disabilities and enable them to read, create, and edit documents with full access to all of the semantics captured by documents creators.

In addition to moral and ethical reasons for creating accessible Office Open XML document, it should be noted that failure to provide to support for users with disabilities can have serious legal implications.

Example of these regulations include:

- Section 508 of the Rehabilitation Act (508) (<http://enable/business/section508.asp>)
- Section 255 of the Telecommunications Act of 1996
- International Organization for Standardization, Part 171 (ISO 171)
- Authoring Tool Accessibility Guidelines (ATAG) (<http://www.w3.org/TR/ATAG10/>)
- Web Content Accessibility Guidelines (WCAG) <http://www.w3.org/TR/WCAG20/>

- User Agent Accessibility Guidelines (UAAG) <http://www.w3.org/TR/UAAG10/>
- Canadian Common Look and Feel (CLF) Guide 1.1 [http://www.cio-dpi.gc.ca/clf-upe/1/1\\_e.asp](http://www.cio-dpi.gc.ca/clf-upe/1/1_e.asp)

There are as well very tremendous business value to create accessible applications. For example, Forrester finds that roughly 57% of computer users are likely or very likely to benefit from the use of accessible technology. Providing specialized support for this community greatly increases potential market size for any Office Open XML implementation.

### Understanding Accessibility

Accessible technology is computer technology which enables individuals to adjust a computer to meet their visual, hearing, dexterity, cognitive, and speech needs. More importantly, accessibility empowers and enables people of all abilities to realize their full potential.

Types of accessible technology include:

- **Accessibility options built into products** such as options that change font size and color.
- **Assistive technology products** which are specialty hardware and software products such as a screen reader or voice recognition product.

While there are laws designed to ensure that businesses do everything possible to maximize accessibility, there are other compelling reasons as well. Businesses today are looking for solutions to empower and retain employees—and accessible technology can help do just that. Accessible technology helps businesses keep great employees, recruit from a larger pool of candidates, and enhance team collaboration and communication among all employees—including those with disabilities.

It's important for businesses to consider the number of people impacted by physical and mental impairments. One study found that 101.4 million working-age adults (ages 18 to 64) in the United States have a mild or severe difficulty/impairment ([Source: Study commissioned by Microsoft, conducted by Forrester Research, Inc. 2003](#)). In addition, large numbers of people experience temporary impairments caused by illness or accident that impact their abilities on the job. The aging of the workforce adds another dimension. Consider that by 2008, 40 percent of the workforce will be comprised of baby boomers that will, in growing numbers, experience physical impairments due to the natural aging process. Providing technology that helps valued employees remain productive makes good business sense.

### Needs by Type of Disability

We can think of disabilities as fitting into one of five buckets (with some needs spanning multiple areas). These include vision, dexterity, hearing, learning, and communication. Below, we provide details on what defines each category and technologies that exist to address needs today.

### Vision

Vision difficulties and impairments include low vision, color blindness, blindness and even conditions such as presbyopia, which is associated with aging populations. Among adult computer users in the United States, 1 in 4 (27%) have a vision difficulty ([see study](#)). There are many options for individuals with vision difficulties to modify the computer display and appearance so it is more legible, or receive information through sound or touch. Those who are blind cannot use a computer monitor but have the option to receive information from their computers through hearing or touch offered through screen readers and Braille displays.

People who have vision difficulties and impairments may be interested in the following assistive technology:

- **Screen enlargers** (or screen magnifiers) work like a magnifying glass. They enlarge a portion of the screen as the user moves the focus—increasing legibility for some users. Some screen enlargers allow a user to zoom in and out on a particular area of the screen.
- **Screen readers** are software programs that present graphics and text as speech. A screen reader is used to verbalize, or "speak," everything on the screen including names and descriptions of control buttons, menus, text, and punctuation.
- **Speech recognition systems**, also called voice recognition programs, allow people to give commands and enter data using their voices rather than a mouse or keyboard.
- **Speech synthesizers** (often referred to as text-to-speech (TTS) systems) receive information going to the screen in the form of letters, numbers, and punctuation marks, and then "speak" it out loud. Using speech synthesizers allows blind users to review their input as they type.
- **Refreshable Braille displays** provide tactile output of information represented on the computer screen. The user reads the Braille letters with his or her fingers, and then, after a line is read, refreshes the display to read the next line.
- **Braille embossers** transfer computer generated text into embossed Braille output. Braille translation programs convert text scanned in or generated via standard word processing programs into Braille, which can be printed on the embosser.
- **Talking word processors** are software programs that use speech synthesizers to provide auditory feedback of what is typed.
- **Large-print word processors** allow the user to view everything in large text without added screen enlargement.

### Dexterity

Individuals with dexterity difficulties experience pain, discomfort, or complete loss of feeling in their fingers, hands, wrists, or arms, making it difficult to use a standard keyboard or mouse. Among adult computer users in the United States, 1 in 4 (26%) have a dexterity difficulty ([see study](#)). Dexterity difficulties and impairments can be caused by a wide range of common illnesses and accidents such as carpal tunnel, arthritis, stroke, cerebral palsy, Parkinson's disease, multiple sclerosis, loss of limbs or digits, spinal cord injuries, and repetitive stress injury, among others.

People who have dexterity difficulties and impairments may be interested in the following assistive technology:

- **Speech recognition systems**, also called voice recognition programs, allow people to give commands and enter data using their voices rather than a mouse or keyboard.
- **On-screen keyboard programs** provide an image of a standard or modified keyboard on the computer screen. The user selects the keys with a mouse, touch screen, trackball, joystick, switch, or electronic pointing device.
- **Keyboard filters** include typing aids, such as word prediction utilities and add-on spelling checkers. These products reduce the required number of keystrokes. Keyboard filters enable users to quickly access the letters they need and to avoid inadvertently selecting keys they don't want.
- **Touch screens** are devices placed on the computer monitor (or built into it) that allow direct selection or activation of the computer by touching the screen.
- **Alternative input devices** (including alternative keyboards, electronic pointing devices, sip-and-puff systems, wands and sticks, joysticks and trackballs) allow individuals to control their computers through means other than a standard keyboard or pointing device.

### Hearing

Hearing difficulties and impairments encompass a wide range of conditions—from slight hearing loss to deafness. People who have hearing difficulties and impairments might be able to hear some sound, but might not be able to distinguish words. Among adult computer users in the United States, 1 in 5 (21%) have a hearing difficulty ([see study](#)).

### Learning Difficulties

Learning difficulties and impairments can range from conditions such as dyslexia and attention deficit disorder to retardation. Processing problems are the most common and have the most impact on a person's ability to use computer applications. These conditions interfere with the learning process. Many individuals with learning difficulties and impairments are of high intelligence and if information is presented to them in a form and at a pace that is appropriate, they are able to be much more productive. During the learning process, many individuals with learning difficulties benefit from having a multisensory experience of audio speech paired with a visual representation. Reducing visual and auditory distractions can also aid the learning process for many people. People who have learning difficulties and impairments may be interested in the following:

- **Word prediction programs** allow the user to select a desired word from an on-screen list located in the prediction window. This list, generated by the computer, predicts words from the first one or two letters typed by the user. The word can then be selected from the list and inserted into the text by typing a number, clicking the mouse or scanning with a switch. These programs help users increase written productivity and accuracy, and increase vocabulary skills through word prompting.
- **Reading comprehension programs** focus on establishing or improving reading skills through ready-made activities, stories, exercises, or games. These programs can help users practice letter sound recognition and can increase the understanding of words by adding graphics, sound, and possibly animation.
- **Reading tools and learning disability programs** include software designed to make text-based materials more accessible for people who struggle with reading. Options can include scanning, reformatting, navigating, or speaking text out loud. These programs help people who have

difficulty seeing or manipulating conventional print materials; people who are developing new literacy skills or who are learning English as a foreign language; and people who comprehend better when they hear and see text highlighted simultaneously.

- **Speech synthesizers** (often referred to as text-to-speech (TTS) systems) receive information going to the screen in the form of letters, numbers, and punctuation marks, and then "speak" it out loud. Individuals who have lost the ability to communicate orally can use a speech synthesizer to communicate by typing information and letting the speech synthesizer speak it out loud.
- **Speech recognition systems**, also called voice recognition programs, allow people to give commands and enter data using their voices rather than a mouse or keyboard.

### Communication

Language and communication difficulties and impairments include conditions such as aphasia (loss or impairment of the power to use or comprehend words, often as a result of brain damage), delayed speech (a symptom of cognitive impairment), and other conditions resulting in difficulties remembering, solving problems, or perceiving sensory information. For people who have these difficulties and impairments, complex or inconsistent visual displays or word choices can make using computers more difficult. People who have language or communication difficulties and impairments may be interested in the following assistive technology:

- **Keyboard filters** include typing aids such as word prediction utilities and add-on spelling checkers. These products reduce the required number of keystrokes. Certain keyboard filters enable users to quickly access the letters they need and to avoid inadvertently selecting keys they don't want.
- **Speech recognition systems**, also called voice recognition programs, allow people to give commands and enter data using their voices rather than a mouse or keyboard.
- **Screen review utilities** make on-screen information available as synthesized speech and pairs the speech with a visual representation of a word, for example, highlighting a word as it is spoken. Screen review utilities convert the text that appears on screen into a computer voice. This helps some people with language difficulties and impairments by giving them information visually and aurally at the same time.
- **Touch screens** are devices placed on the computer monitor (or built into it) that allow direct selection or activation of the computer by touching the screen.
- **Speech synthesizers** (often referred to as text-to-speech (TTS) systems) receive information going to the screen in the form of letters, numbers, and punctuation marks, and then "speak" it out loud.

## Creating Accessible Applications with Office Open XML

### The Role of the Content Author

Even the most accessible Office Open XML implementation will require some level of engagement from the content author. Any accessible implementation must ensure, at a bare minimum, that some UI is available to the author for providing key accessible elements like alternative text for images. Clearly, there are different lengths to which a developer could go to provide this experience. At the most

limited end of the spectrum an implementation could provide nothing but the option to provide this information; however, an extremely accessible implementation could include things like a “checker” that parses a document for lack of information like alternative text, prompts the user, and provides instructions as to what should be provided.

In the end, the author must be provided with the necessary UI to ensure they can include the support described below. Failure to provide a straightforward way in which to provide this information and structure is likely to result in what is only a partially accessible Office Open XML implementation.

### Accessibility Engineering

**Product UI controls and content should be accessible.** This includes electronic forms, UI controls, self contained software products, scripts, applets, plug-ins, Component Object Model (COM) objects, and all other programmatic elements that generate or interpret page content.

**Product UI controls and content should be compatible with assistive technologies.** A user can thus employ assistive technology, such as screen readers, magnifiers, voice recognition, and alternative input devices to help them use the product. However, if the product is a self-contained product such as a kiosk or cell phone, for example, it should not require that such a user add assistive technology in order to use the product.

**If a product is an authoring tool, it should enable an author to edit the structure of a document,** as well as all of the properties of each element and object, in an accessible fashion.

**A product should address the needs of people with vision difficulties, hearing difficulties, cognitive difficulties, or motor difficulties.** Specifically, the implementation should provide at least one mode of operation and information retrieval that does not require that a user be able to see, hear, speak, move with fine motor control, do two things at once, reach, or exert physical strength. Except in the case of a self-contained product, the product may accomplish this by facilitating the use of assistive technology developed by third-party vendors.

### Navigation

There numerous things a developer must consider with respect to navigation in order to create an accessible Office Open XML implementation. At a high level these include the following:

1. It must be possible to do everything using only the keyboard. There should be no functions that are dependent on the input of a mouse or other similar device.
2. It must also be efficient to do everything via the keyboard. For example, it is unreasonable to require keyboard users to press TAB to navigate through hundreds of controls, just to get to the control that interests them.
3. It must be possible to do everything using only one hand. Shortcut key sequences should be reasonable to perform with one hand.
4. Keyboard navigation should be consistent and reasonable. TAB should navigate from one control to another. ESC should cancel the current keyboard state. The most commonly used

controls should be visited first in the TAB order. If there are numerous controls laid out in two dimensions, the arrow keys should also work for two-dimensional keyboard navigation.

### Keyboard Focus

Users must know which object has the keyboard focus at anytime. Avoid confusion by hiding all visual focus indicators and dimming selections that are located in inactive windows (or panes). To highlight the keyboard focus, use colors, fonts, graphics such as rectangles, or magnification. Highlight the keyboard focus audibly by changing the volume, the pitch, or the tonal quality. In addition an accessible Office Open XML implementation should ensure that:

- An object always has the keyboard focus, and the keyboard focus is always visible and obvious. When the focus moves to another element, the new position is obvious.
- If a UI control currently has the focus, it is visually highlighted.
- All selections are visually highlighted.
- The keyboard focus is displayed independently of a selection, allowing support for multiple and noncontiguous selections.
- The implementation can show a selection that exists in an inactive window or pane; this can be done, for example, by dimming the selection.
- Only one keyboard focus indicator shows at any one time.
- Users can place the focus anywhere within the textual content or on any UI control with which they can interact.

### Existing Navigation Guidelines

An accessible Office Open XML implementation must follow the keystroke guidelines provided by the OS on which the application is built. Guidelines exist for most major operating systems including Windows, Macintosh, and Linux.

### Themes and Contrast Support

Users often have unique demands of their computing environment (e.g. high contrast by default) and the settings associated with these demands must be followed by any accessible Office Open XML implementation. System-wide settings should never be disabled or disregarded by a product if they are intended to increase accessibility. Many users require specific high-contrast combinations, such as white text on a black background. Drawing these reversed, as black text on a white background, causes the background to bleed over the foreground and can make reading difficult, even painful, for some users. Text is most legible for everyone when drawn on a plain background of a contrasting color. Text drawn over a variegated background, such as a wash of colors or a bitmap, can be difficult to read for the person without disabilities, and can be unreadable for users who have turned on a high contrast-like mode.

### Assistive Technologies (AT)

Any accessible Office Open XML application must provide at least one mode that allows for full use of the product without the use of vision or by interacting directly with the users' assistive technologies.

Failure to provide at least one of these options will make it so that certain users have no means by which to use an application.

### Programmatic Access

A programmatic accessibility technology is fundamentally a method of providing information to the AT through some publically defined API (or set of API's). Assistive Technologies must be able to identify and manipulate the elements of an application's user interface. The standard mechanisms of an application should ensure that AT has access to any content presented textually or through other visual representations on the screen. An accessible Office Open XML implantation should do this by:

- Providing programmatic access through one of the many accessibility frameworks available through operating systems. An example of this would be UI Automation. UI Automation provides programmatic access to most user interface (UI) elements on the desktop, enabling assistive technology products such as screen readers to provide information about the UI to end users and to manipulate the UI by means other than standard input. UI Automation also allows automated test scripts to interact with the UI. UI Automation client applications can be written with the assurance that they will work on multiple frameworks. The UI Automation core masks any differences in the frameworks that underlie various pieces of UI. For example, the **Content** property of a WPF button, the **Caption** property of a Win32 button, and the **ALT** property of an HTML image are all mapped to a single property, **Name**, in the UI Automation view.
- Other methods such as the use of a document object model (DOM). A DOM can be defined as the document-level information source. For example, the Document Object Model (DOM) as implemented in MSXML provides a programmatic representation of XML documents, fragments, nodes, or node-sets. It also provides an application programming interface for working with XML data. As an XML representation, it conforms to the W3C DOM specification. As a set of APIs, XML DOM objects are COM objects that implement interfaces and can be used in XML applications written in programming languages such as C/C++, Visual Basic, VBScript, and JScript.

### Additional Programmatic Access

The following guidelines should be followed to ensure an accessible Office Open XML specification:

- The system caret (the blinking vertical bar that users see when editing text) exposes the location of the keyboard focus for assistive technology. The system caret can be placed anywhere on the screen and made any shape or size. It can also be made invisible and moved to indicate the keyboard focus location for the benefit of other applications without disturbing what the user sees on the screen. The system caret must be sized and positioned to cover the screen element's bounding rectangle, so that screen magnification tools can zoom in on any part of the complete object. This helps the user look for an object's label or judge its size, and it enables utilities to highlight the object that has the keyboard focus.
- The application must provide programmatic access to all textual content and all text attributes.

- The application associates a unique, meaningful title (and a description, if needed) with frames, objects, windows, and Web pages to facilitate identification and navigation.
- The description associated with a frame, object, window, or Web page gives the purpose of the item and explains how that item relates to other items. (A description is necessary only if the purpose and relationship is not clear from the title alone.)
- Vector drawings and elements have `title` and `desc` tags. The alternative text is reused as graphics.
- The application must provide a mechanism by which the user can view headers (row and column) for each cell in a table.

### Application Help

The help system of an Office Open XML system must also be accessible. In addition, developers should do everything possible to make access to a help system as easy possible. Advanced methods for achieving this include providing direct access to help topics via any features in the UI. Searchable help systems tend to be more useful to users with disabilities.

## Document Level Accessibility

Accessibility at the application level in of itself is not sufficient to produce a fully accessible Office Open XML implementation. In order for users with disabilities to have complete access to document content there are numerous which must be taken into account. Moreover, the aspects called out below are the components of a document that should be maintained when converting between Office Open XML and other formats such as HTML and ODF.

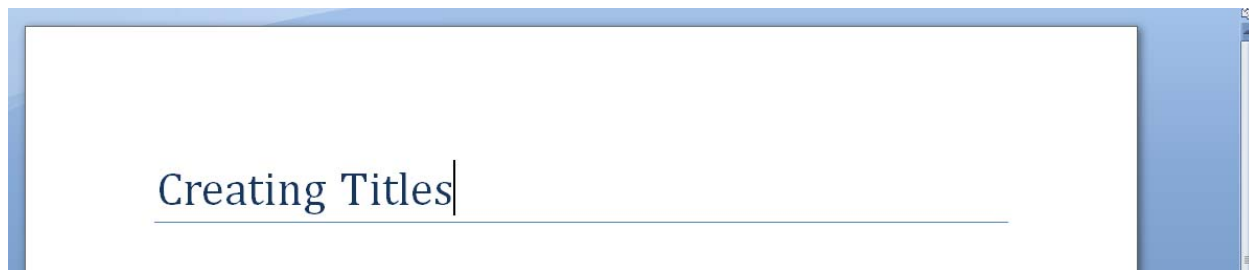
### Document Basics

An accessible Office Open XML implementation should be capable of producing documents with a structure that is easily navigated and understood. At the most elementary level this is simply matter of good content design and is a role that is owned by the document author; however, the use of properly-defined styles, themes, and so on is critical. Consider the following logical document structure:

- I. Title Page with Abstract
- II. Introduction
- III. Table of Contents
- IV. Primary Content of Document
- V. Citations
- VI. Appendix

An unacceptable example of a title would be merely increasing the defined font size and giving it a bold attribute. Instead, a title should be properly styled according to the Office Open XML specification. In WordprocessingML, titles are a combination of run-level and paragraph level formatting. Run-level formatting provides the ability to assign characteristics to individual characters. This includes characteristics like titles, italics, and bold type font. The other level of formatting is at the paragraph level and includes styles like title.

Below is a sample title as it would appear in the finished document:



Below, we define the run-level formatting that provides the visual appearance of title.

```
<w:p>
  <w:r>
    <w:rPr>
      <w:b />
      <w:sz w:val = "52" />
      <w:rFonts w:ascii="Cambria" />
    </w:rPr>
    <w:t>Creating Titles</w:t>
  </w:r>
</w:p>
```

However, for users with disabilities, this appearance is not enough. Instead, an accessible Office Open XML implementation must also take into account the paragraph level formatting and the title style.

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Title" />
  </w:pPr>
  <w:r>
    <w:t>Creating Titles</w:t>
  </w:r>
</w:p>
```

Providing properly defined styles is critical for access technologies which can then provide this information to users. A user can quickly determine the nature and structure of content and in turn, is provided with an acceptable consumption experience. If the title is above is not defined properly (for example, just made bold) then a user with disabilities may miss out on key information or the meaning of a document may be drastically changed.

All accessible Office Open XML output should consist of a document following a logical navigation and structural order while also providing properly defined styles throughout.

### Styles

Styles must be utilized correctly to create a completely accessible Office Open XML implementation. Styles form the foundation of any document and if used correctly can greatly aid in providing succinct and accurate information to AT and other tools.

For example, in the Office Open XML specification, styles are stored in the WordprocessingML package in the styles-part. This package contains WordprocessingML specific XML markup. Within this package there are three elements including styles, latent-styles, and document defaults. All of these style types, which are defined in more depth in the glossary, can be used to define formatting throughout a document.

### Tables

Tables can be very difficult to understand for users with disabilities if those tables are not designed properly. In order to ensure that tables are accessible, authors must take into consideration two topics, table navigation and table labeling. Moreover, authors must ensure that they preserve multiple elements associated with a table including: captions/alternative text, equations, column and row header information. Considerable work was done in the Office Open XML specification to ensure that users with disabilities would be able to readily consume information in tables. Table accessibility is described in detail below.

For example, when a user creates an Office Open XML document, uses tables, and marks the header to repeat on all the pages, table styles and width and table header rows can be specified. The element `<w:tblHeader/>` tag, represents whether the row is a header (which repeats on all pages) or not.

```
<w:tbl>
  <w:tr >
    <w:trPr>
      <w:tblHeader/>
    </w:trPr>
    <w:tc>
  ...
  </w:tc>
  <w:tc>
  ...
  </w:tc>
  </w:tr>
</w:tbl>
```

### headers (Header Cells Associated With Table Cell)

This element specifies the list of header cells, as specified by children header elements, that provide header information associated with the current table cell. Each header cell must specify a unique identifier as specified by the use of the attribute `id` on the header cell `tc` element. This element is typically used to gather header information about data and sub header cells.

If this element is omitted or there exist no children header elements, then no header cell shall be associated with the given table cell.

[*Example:* Consider the following 3 x 3 table with four header cells having values A, B, C, and D, and four data cells with values x1, x2, y1, and y2:

|   |    |    |
|---|----|----|
|   | A  | B  |
| C | x1 | x2 |
| D | y1 | y2 |

Each of the data cells is associated with two header cells and can be represented in WordProcessingML as follows:

```
<w:tbl>
  <w:tr>
    <w:tc >
      </w:tc>
    <w:tc w:id="HeaderA">
      <w:p>
        <w:r>
          <w:t>A</w:t>
        </w:r>
      </w:p>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc w:id="HeaderC">
      <w:p>
        <w:r>
          <w:t>C</w:t>
        </w:r>
      </w:p>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:headers>
          <w:header w:val="HeaderA" />
          <w:header w:val="HeaderC" />
        </w:headers>
      </w:tcPr>
      <w:p>
        <w:r>
          <w:t>x1</w:t>
        </w:r>
      </w:p>
    </w:tc>
  </w:tr>
</w:tbl>
```

The headers element specifies the list of header cells associated with the table cell that has a value of x1. In this example x1 is associated with headers that have an id of HeaderA and HeaderC.

### **header (Header Cell Reference)**

This element specifies a reference, using a unique identifier, to a table header cell that is associated with the current table cell. The identifier representing the reference shall be stored on this element's `val` attribute and is used to reference the unique identifier value of a table header cell. The contents of the table header cell resolved by a specific unique identifier shall be used as the table header information associated with the table cell that references that specific unique identifier.

### **Alternative Text**

Alternative text (alt text) is the text that is used to describe things like drawings, images, and charts. Office Open XML has two attributes that are used in the specification to represent alternative text (depending on type of object). These are “`descry`” and “`alt`”. Accessible Office Open XML implementations must provide support for alt text and authors should ensure that whenever any of the objects defined below are utilized, alt text is provided. Provided below are samples of alt text for different objects that would be used in Office Open XML. One of the tremendous benefits of the format is that any object can be populated with alternative text, greatly increasing document accessibility.

At this time Office Open XML does not support long and short alternative text, but it something that will be evaluated for future iterations of the specification.

### Image

```
<w:drawing>  
<wp:docPr id="1" name="Picture 0" descr="A tree with many leaves falling in autumn"/>  
</w:drawing>
```

### Shapes

```
<v:shape id="_x0000_s1026" type="#_x0000_t202" alt="Text Box which has text"></v:shape>
```

### Charts

```
<w:drawing>  
<wp:docPr id="1" name="Chart 1" descr="This chart gives information about Category and  
Series"/>  
</w:drawing>
```

### Grouped Objects

```
<v:group id="_x0000_s1029" alt="This group of Shapes has a Red Rectangle, Blue Ellipse and  
Green Triangle." ...>  
</v:group>
```

### Line and Arrow

```
<v:shape id="_x0000_s1026" type="#_x0000_t32" alt="this is a red line"/>
```

### 3D Objects

```
<w:drawing>  
  <wp:docPr id="2" name="Diagram 2" descr="This is a 3D View"/>  
</w:drawing>
```

## Headings

Office Open XML supports headings and multiple heading levels. Like other critical components of document structure, these must be supported and utilized to ensure an accessible Office Open XML implementation. An example is provided below. For example, when Header elements (like, Heading 1, Heading 2 and other kind of styles) are used in an Office Open XML document, each of the paragraphs is provided with the respective style. The length of the XML fragment increases due to the non-mixed content model and run-of-text formatting.

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Title" />
  </w:pPr>
  <w:r>
    <w:t>Title</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pStyle w:val="Heading1" />
  </w:pPr>
  <w:r>
    <w:t>Heading 1</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pStyle w:val="Heading2" />
  </w:pPr>
  <w:r>
    <w:t>Heading 2</w:t>
  </w:r>
</w:p>
```

## Lists

Non-visual users may "get lost" in lists, especially in nested lists and those that do not indicate the specific nest level for each list item. Hence proper mark-up should be provided for the List and list-items; this also helps for better navigation in the document. This is supported by Office Open XML.

When nested lists are used in an Office Open XML document, a new xml part named "numbering.xml" is generated and the association is referenced in the "document.xml" as a paragraph property thru the <w:numPr> element. This will have the list level with the appropriate identifier. The start element in the abstract numbering definition handles the identifier.

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="ListParagraph"/>
    <w:numPr>
      <w:ilvl w:val="0"/>
      <w:numId w:val="2"/>
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>One</w:t>
  </w:r>
</w:p>
<w:p >
  <w:pPr>
    <w:pStyle w:val="ListParagraph"/>
    <w:numPr>
      <w:ilvl w:val="1"/>
      <w:numId w:val="2"/>
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Cat</w:t>
  </w:r>
</w:p>
```

## Headers and Footers

It is essential to ensure that header and footer information is preserved correctly. This allows users with disabilities to navigate throughout components of a page with fewer barriers. In Office Open XML this information is provided as shown below.

**Header -**

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Header"/>
  </w:pPr>
  <w:r>
    <w:t>Test</w:t>
  </w:r>
</w:p>
```

**Footer -**

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Footer"/>
  </w:pPr>
  <w:r>
    <w:t>bill.jones</w:t>
  </w:r>
</w:p>
```

## Form Elements

When working with form elements in the Office Open XML formats there are two components that must be taken into consideration. They are labels for form controls and titles for form controls. The Office Open XML formats provide explicit support for these elements. This is described below.

### label (Structured Document Tag Label)

This element specifies the label identifier associated with the current structured document tag. The identifier representing the label shall be stored on this element's `val` attribute and is used to reference the unique identifier value of a structured document tag. The contents of the structured document tag resolved by a specific unique identifier shall be used as the label content for the structured document tag that references that specific unique identifier of the structured document tag.

If this element is omitted or the value of the label identifier cannot be resolved, then no label shall be associated with the given structured document tag.

Consider the following two structured document tags where one structured document tag references another structured document tag as a label:

```
<w:sdt>
  <w:sdtPr>
    <w:id w:val="5" />
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>Name</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
...
<w:sdt>
  <w:sdtPr>
    <w:id w:val="6" />
    <w:label w:val="5" />
  </w:sdtPr>
  ...
</w:sdt>
```

The label element specifies that the structured document tag with an identifier value of 6 will use the contents of the structured document tag with an identifier value of 5 as a label source. In this example, the label contents will be “Name”.

### **label (Form Field Label)**

This element specifies the label identifier associated with the current form field. The identifier representing the label shall be stored on this element’s val attribute and is used to reference the unique identifier value of a structured document tag. The contents of the structured document tag resolved by a specific unique identifier shall be used as the label content for the form field that references that specific unique identifier of the structured document tag.

If this element is omitted or the value of the label identifier cannot be resolved, then no label shall be associated with the given form field.

Consider the following text box form field, which references a structured document tag as a label:

```
<w:sdt>
  <w:sdtPr>
    <w:id w:val="5" />
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>Name</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
...
<w:ffData>
  <w:name w:val="TextBox" />
  <w:enabled />
  <w:textInput />
  <w:label w:val="5" />
</w:ffData>
```

The label element specifies that the text box form field will use the contents of the structured document tag with an identifier value of 5 as a label source. In this example, the label contents will be “Name”.

## Page and Section Breaks

In documents it is often critical to maintain information regarding page and section breaks. Being able to navigate by page and section must be possible for users with disabilities. This element is also required for creating the DAISY XML, which is described later via the Office Open XML-to-DAISY translator.

WordprocessingML does not natively store the concept of pages, since it is based on paragraphs and runs (which are laid out on to pages by consumers of this content).

To provide clarity, a section is denoted by the sectPr element, its child element “type” tells you whether the section starts on the first even page, odd page, next page, and so on. A “hard” page break is noted in the br element within the paragraph and its type attribute tells you the break type. For example:

```
<w:r>
  < w:br w:type="page" />
</w:r>
```

Consider a document with four paragraphs of text that is to be printed on a page in landscape mode, followed by ten paragraphs of text that are to be printed in portrait mode. This requirement implies information about the page(s) used to lay out each grouping of text—the first four paragraphs could require one page, or ten. Therefore, rather than try to cache knowledge of the number of pages and their properties (which is likely to become invalid if the XML is manipulated by a producer that does not understand page layout), this information is stored by breaking the document into two sections, as follows:

```
<w:p>
...
</w:p>
<w:p>
...
</w:p>
<w:p>
...
</w:p>
<w:p>
<w:sectPr>
...
<w:pgSz ... w:orient="landscape" />
...
</w:sectPr>
...
</w:p>
...
<w:p>
<w:sectPr>
...
<w:pgSz ... w:orient="landscape" />
...
</w:sectPr>
...
</w:p>
```

### Logical Navigation

Accessible Office Open XML implementations must provide support for logical navigation order. Specifically, users must be able to explicitly specify the navigation order among elements on things like a presentation slide or between objects in various content types. The Office Open XML specification was engineered to provide support specifically for this need.

### **tabIndex (Structured Document Navigation Order Index)**

This element specifies the position of the current structured document tag in the navigation (tab) order used in the document. The index shall be stored on this element's val attribute and is analogous to the tabIndex attribute in HTML.

Objects that support tab index shall be navigated by consumers in the following order:

Objects for which the XML specifies a non-zero tabIndex value are navigated first. Navigation proceeds with the element with the lowest resolved value of tabIndex to the element with the highest resolved value of tabIndex.

Objects that specify identical resolved values of tabIndex will be navigated in the lexical order in which the elements appear in the underlying WordprocessingML.

Objects for which the XML does not specify an index or objects for which the XML specifies a resolved tabIndex value of 0 are navigated last. These objects are navigated in the lexical order in which they appear in the underlying WordprocessingML. Consider the following two structured document tags where each structured document tag specifies tab index:

```
<w:sdt>
  <w:sdtPr>
    <w:id w:val="5" />
    <w:tabIndex w:val="1" />
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>First Name</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
...
<w:sdt>
  <w:sdtPr>
```

```
<w:id w:val="6" />
<w:tabIndex w:val="2" />
</w:sdtPr>
<w:sdtContent>
  <w:p>
    <w:r>
      <w:t>Last Name</w:t>
    </w:r>
  </w:p>
</w:sdtContent>
</w:sdt>
```

The `tabIndex` element specifies that the structured document tag with an identifier value of 5 shall be the first content to be reached via tabbing, whereas the structured document tag with an identifier value of 6 shall be the second content to be reached via tabbing.

#### **tabIndex (Form Field Navigation Order Index)**

This element specifies the position of the current form field in the navigation (tab) order used in the document. The tabbing index shall be stored on this element's `val` attribute and is analogous to the `tabIndex` attribute in HTML.

Objects that support tab index shall be navigated by consumers in the following order:

Objects for which the XML specifies a non-zero `tabIndex` value are navigated first. Navigation proceeds with the element with the lowest resolved value of `tabIndex` to the element with the highest resolved value of `tabIndex`.

Objects that specify identical resolved values of `tabIndex` will be navigated in the lexical order in which the elements appear in the underlying WordprocessingML.

Objects for which the XML does not specify an index or objects for which the XML specifies a resolved `tabIndex` value of 0 are navigated last. These objects are navigated in the lexical order in which they appear in the underlying WordprocessingML.

Consider the following two text box form fields where form field specifies a tab index:

```
<w:ffData>
  <w:name w:val="FirstName" />
  <w:enabled />
  <w:textInput />
  <w:tabIndex w:val="1" />
</w:ffData>
...
<w:ffData>
  <w:name w:val="LastName" />
  <w:enabled />
  <w:textInput />
  <w:tabIndex w:val="2" />
</w:ffData>
```

The tabIndex element specifies that the FirstName form field shall be the first content to be reached via tabbing, whereas the LastName form field shall be the second content to be reached via tabbing.

Each shape-based object within the shape tree, whether grouped or not, shall represent one unique level of z-ordering on the slide. The z-order for each shape-based object shall be determined by the lexical ordering of each shape-based object within the shape tree: the first shape-based object shall have the lowest z-order, while the last shape-based object shall have the highest z-order.

The z-ordering of shape-based objects within the shape tree shall also determine the navigation (tab) order of the shape-based objects: the shape-based object with the lowest z-order (the first shape in lexical order) shall be first in navigation order, with objects being navigated in ascending z-order.

Consider the following PresentationML slide with two shapes

```
<p:sld>
  <p:cSld>
    <p:spTree>
      ...
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="5" name="Oval 4" />
          ...
        </p:nvSpPr>
      ...
    </p:sp>
    <p:sp>
      <p:nvSpPr>
        <p:cNvPr id="4" name="Isosceles Triangle 3" />
        ...
      </p:nvSpPr>
    ...
  </p:spTree>
</p:cSld>
...
</p:sld>
```

In the above example, the shape with name Oval 4 has the lowest z-order value since that shape is the first shape in the shape tree. Oval 4 is also the first shape in navigation order. The shape with name Isosceles Triangle 3 has the highest z positioning value since that shape is the last shape in the shape tree. Isosceles Triangle 3 is also the last shape in navigation order.

## Frames

Frames should be titled with text that facilitates identification and navigation. The two key considerations for frames and framesets are:

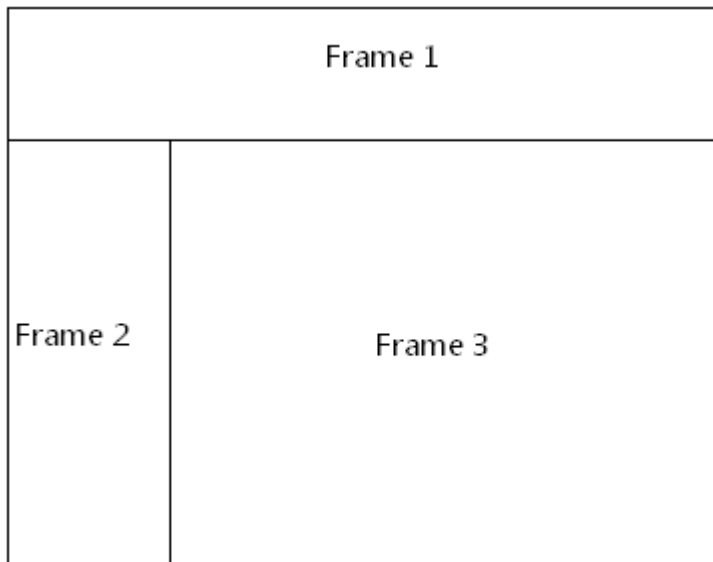
1. Each frame and frameset must contain a title to facilitate frame identification and navigation.
2. The purpose of frames as well as how frames relate to each other should also be given, if it not obvious by the frame titles alone.

**title (Frame or Frameset Title)**

This element specifies advisory information about a single frame or frameset. The title information shall be stored in this element's val attribute. This property is analogous to the title attribute on the frame or frameset element in HTML.

If this element is omitted, then no title shall be associated with the given frame or frameset.

Consider a WordprocessingML document that serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:title w:val="Our library of documents" />
  <w:frame>
    <w:name w:val="Frame 1" />
    <w:title w:val="Menu bar" />
  </w:frame>
</w:frameset>
<w:frameset>
  <w:title w:val="Navigation and document collection" />
  ...
  <w:frame>
    <w:name w:val="Frame 2" />
    <w:title w:val="Navigation bar" />
  </w:frame>
  <w:frame>
    <w:name w:val="Frame 3" />
    <w:title w:val="Documents" />
  </w:frame>
</w:frameset>
</w:frameset>
```

The title element specifies supplementary information for each frame and frameset. In this case, the frames have titles of “Menu bar”, “Navigation bar”, and “Documents”, respectively, while the framesets have titles of “Our library of documents”, and “Navigation and document collection”, respectively.

**longDesc (Frame Long Description)**

This element specifies a link to a long description of the frame. This description should supplement the short description provided by the title element. This property is analogous to the longdesc attribute on the frame element in HTML.

If this element is omitted, then no long description shall be associated with the given frame.

The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
...
<w:frame>
  <w:name w:val="Frame 1" />
  <w:title w:val="Menu bar" />
  <w:longDesc r:id="rIdMenuBar" />
</w:frame>
<w:frameset>
  <w:title w:val="Navigation and document collection" />
  ...
  <w:frame>
    <w:name w:val="Frame 2" />
    <w:title w:val="Navigation bar" />
    <w:longDesc r:id="rIdNavBar" />
  </w:frame>
  <w:frame>
    <w:name w:val="Frame 3" />
    <w:title w:val="Documents" />
    <w:longDesc r:id="rIdDocs" />
  </w:frame>
</w:frameset>
</w:frameset>
```

The longDesc element specifies that the part targeted by the relationship with an id of rIdMenuBar shall be used for supplementary information for Frame 1. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

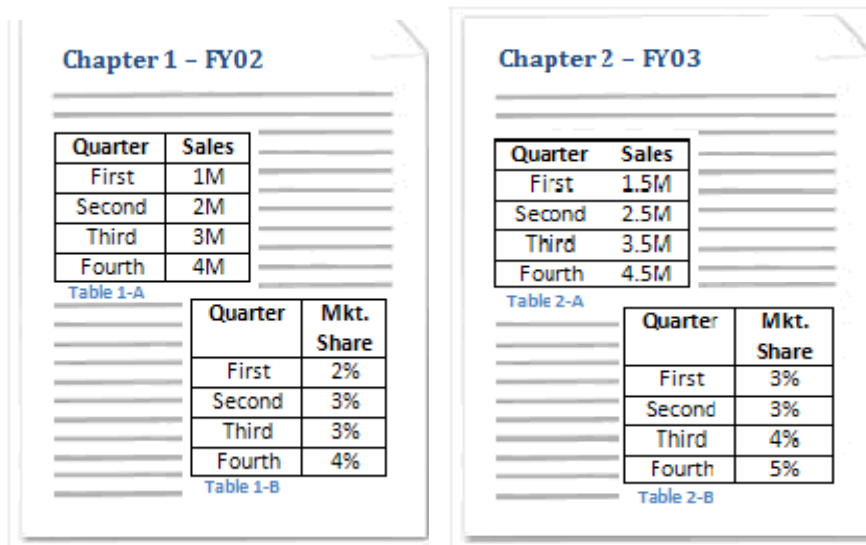
```
<Relationships ... >
...
<Relationship Id="rIdMenuBar" TargetMode="External"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink" Target="myexample-desc.html#menubar" />
...
</Relationships>
```

The corresponding relationship part item shows that the long description supplementary information for Frame 1 is located at myexample-desc.html#menubar, where myexample-desc.html describes #menubar as “This frame provides links to the major sections of the site: Home, Resources, Links, Help.”

## Captions

### autoCaption (Single Automatic Captioning Setting)

This element specifies what type(s) of objects shall automatically labeled with captions, and with which captions the specified objects shall be labeled as defined in the caption element. Consider the diagram below illustrating a two page WordprocessingML document that has leveraged WordprocessingML to automatically label WordprocessingML tables with a specified caption when tables are inserted into the given document.



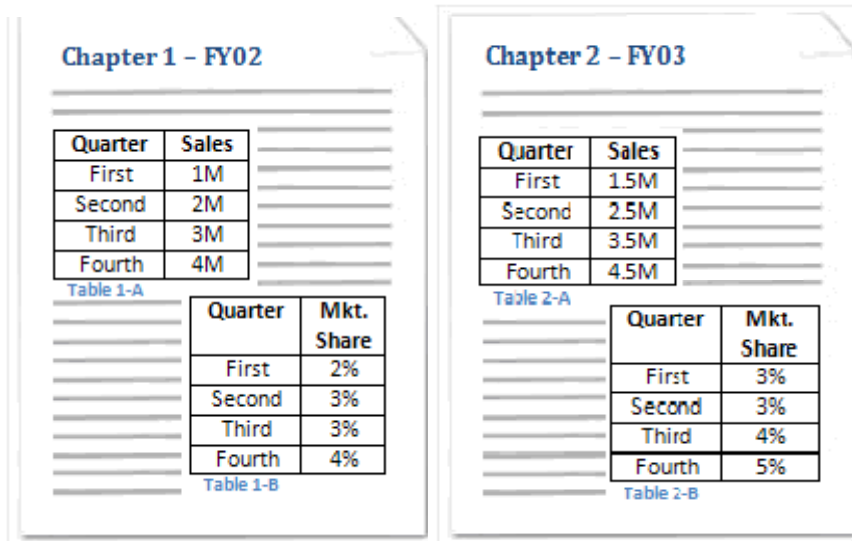
This type of automatic captioning is specified using the following WordprocessingML fragment:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:chapNum="On"
w:heading="2" w:numFmt="upperCase" w:sep="8212" />
  <w:autoCaptions>
    <w:autoCaption w:name="wfwTable" w:caption="Table" />
  </w:autoCaptions>
</w:captions>
```

Here, the autoCaption element specifies through the name attribute being set equal to wfwTable that tables will automatically be labeled with the caption specified in the caption element whose name attribute is equal to Table, as the caption element's caption attribute has a value of Table.

### autoCaptions (Automatic Captioning Settings)

This element specifies that one or more types of objects, when inserted into a WordprocessingML document, will automatically be labeled with a specific caption defined using the caption element. Consider the following example illustrating a two page WordprocessingML document that has leveraged WordprocessingML to automatically label WordprocessingML tables with a specified caption.



This type of automatic captioning is specified using the following WordprocessingML fragment:

```
<w:captions>  
  <w:caption w:name="Table" w:pos="below" w:chapNum="On" w:heading="2"  
w:numFmt="upperCase" w:sep="8212" />  
  <w:autoCaptions>  
    <w:autoCaption w:name="wfwTable" w:caption="Table" />  
  </w:autoCaptions>  
</w:captions>
```

The autoCaptions element specifies set of objects that when inserted into a WordprocessingML document will automatically be labeled with a given caption.

## Language

When natural language changes are marked up in a document, speech synthesizers and Braille devices can automatically switch to the new language, making the document more accessible to multilingual users. For example, when an Office Open XML document uses different languages for text, a clear identification of the language used and its country is provided. The <w:lang/> tag is generated which indicates the language used in the paragraph and run.

```
<w:p>
  <w:r w:rsidR="0008256F">
    <w:t xml:space="preserve">multiple languages like </w:t>
  </w:r>
  <w:r w:rsidR="0008256F">
    <w:t>شقشلا هؤ سشعبيه شقشلا هس</w:t>
  </w:r>
  <w:r w:rsidR="0008256F">
    <w:rPr>
      <w:rFonts w:hint="cs"/>
      <w:rtl/>
      <w:lang w:bidi="ar-DZ"/>
    </w:rPr>
    <w:t>شقشلا هؤ شملتق هس</w:t>
  </w:r>
  <w:r w:rsidR="0008256F">
    <w:rPr>
      <w:rFonts w:hint="cs"/>
      <w:rtl/>
      <w:lang w:bidi="ar-EG"/>
    </w:rPr>
    <w:t>شقشلا هؤ ثلغحف</w:t>
  </w:r>
</w:p>
```

### **DAISY and Office Open XML**

In addition to the support described above, considerable work has been done to ensure that Office Open XML can be translated into other formats. Specifically, a translator project has been initiated on SOURCEFORGE.net® that will allow for the translation of Office Open XML to the DAISY DTBook format. The architecture was created in a way that allows for the translator to be easily implemented easily across various operating systems and applications. More information on this project can be found at

<http://sourceforge.net/projects/openxml-daisy>