Accessibility Guidelines for Implementations of Open Document Format v1.1

Draft 14, 2 November, 2006

Document identifier:

ODF_Accessibility_Guidelines_14_2Nov06.odt

Location:

This Version: http://www.oasis-open.org/committees/office
Previous Version: http://docs.oasis-open.org/office/v1.0

Editors:

Peter Korn, Sun Microsystems, Inc. & Rich Schwerdtfeger, IBM
co-chairs of the OASIS OpenDocument Accessibility Subcommittee

Authors:

Janina Sajka, Capital Accessibility
Stephen Noble, Design Science, Inc.
Chieko Asakawa, IBM
Pete Brunet, IBM
Rich Schwerdtfeger, IBM
Dave Pawson, Royal National Institute for the Blind
Peter Korn, Sun Microsystems, Inc.

Abstract:

This document is a guide for Office Applications, that support version 1.1 of the OpenDocument format, to promote and preserve accessible ODF documents. This guide is not a comprehensive guide for content mapping to platform accessibility APIs.

Status:

This document is an early draft.

Table of Contents

1 Background and Overview

1.1 Introduction

The Open Document Format v1..1 (ODF) is capable of encoding and persisting a tremendous amount of structural and semantic information needed by people with disabilities and the tools they use (assistive technologies) to gain access to computers and information. This document describes the things that an ODF 1.1 implementation must do in order to gain, persist, and present all of the information that ODF 1.1 is capable of persisting, so that users with disabilities are equally able to read, create, and edit documents – with full access to all of the meaning and intent – as their mainstream colleagues.

1.2 What is Accessibility?

Accessibility is about enabling people with disabilities to participate in substantial life activities that include work and the use of services, products, and information. In the context of ODF documents, this means that people with disabilities should be able to participate fully in the creation, review, and editing process. A blind person, for example, should be able to work with a document someone else created (including getting a text description of an image). A person without the use of hands should be able to fill out a form. Someone with very poor vision should be able to read through your presentation.

1.2.1 Types/Categories of Access

There are two or three different “types or “categories of access. The first is *direc*t access. With direct access, the person with the disability is capable direct interaction . For example, a person who is deaf has direct access to a newspaper – their disability isn't impacted by the media and medium they are accessing. A person who is blind (but not deaf) can use a phone directly.

The second category of access is *mediated* access, or access through an assistive technology. Here the person with the disability is interacting with the media/medium via some other tool. A person with poor vision might use a magnifying glass to read a book (but note: that same person might have direct access to a large print book). Someone who is deaf might enjoy a television show via close captioning (one could argue this is really direct access; one can split hairs).

The third category of access isn't really access. A friend reading the mail to the blind recipient might fall into this category. Could we call this indirect access?

In the computer realm, direct access is that which the program does itself. The large print theme of the desktop and applications for someone with some vision

impairment is the equivalent of the large print book. For someone who is blind, a special program called a *screen reader* is used which mediate the user's experience, reading to them via speech (or rendering via a refreshable Braille display) the contents of the screen. And for the user whose vision is poor enough that the large print theme doesn't work (but still has some vision), a *screen magnifier* may be an option, which magnifies the contents of the screen.

It is worth noting that many of the things we have built specifically for accessibility for people with disabilities are used and appreciated by people without disabilities. Close captioning of television broadcasts has become very popular in noisy places like bars and airports; while captioning in multimedia presentations has allowed text searches of those presentations. Similarly, the wheelchair ramps that allow people in wheelchairs to easily move from the street to the sidewalk are much appreciated by delivery personnel, parents with strollers, and anyone else on wheels (bicycles, skateboards).

1.2.2 Types of Disabilities, and How they are Addressed

There are several different types of disabilities, and they are addressed (or adapted to) with either direct or mediated access techniques. These types of disabilities mostly stem from impairments in one or more of the primary senses.

1.2.2.1 Minor vision impairments

People with minor vision impairments have one or more problems that can be addressed via direct access techniques. These include some level of color blindness (roughly 10% of the male population worldwide has some level of color blindness), the inability to see text below 15 or 18 or 24 point text (an issue for most people as they age), or some diminution of their field of view that isn't very severe. To be considered a true impairment, it must be something that hasn't otherwise been fully addressed by corrective lenses such as glasses.

These users need the default presentation of the ODF application and ODF content rendered somewhat differently. For example, they might need a particular color scheme, or a special high-contrast scheme. They might need the size of the text and images to be larger. The specific techniques ODF applications must employ for users with these needs are described in section <reference to current section 2.2 – need some sort of automatic reference here [[[FIXME]]]>

1.2.2.2 Major vision impairments

People with major vision impairments need more adaptation in the presentation of an ODF document than simply rendering the text in an alternate color scheme, or with simply a larger font. These users need the entire presentation magnified – typically between 2 and 16 times (one pixel becoming 4, or 9, or 16, all the way

up through 256 pixels). Such a significant level of magnification cannot be accommodated within the boundaries of most computer screens. Instead, a second piece of software – an assistive technology called a *screen magnifier* - is used to render a magnified image to a virtual screen that is far larger than the physical screen. Then, this screen magnifier software displays only the appropriate portion of the magnified screen on the physical screen: the portion that the user is interacting with at that moment. In addition, this screen magnifier may be magnifying an original, scaled image that is in a particular color scheme; or the magnifier itself may alter the colors. Some of the more sophisticated screen magnifiers offer additional features, such as reading the text they are magnifying via synthetic speech; panning and zooming through the text contents (so a user doesn't need to move the mouse to read and review the screen); or even extracting the ODF text contents and re-rendering it in a different font, in a specialized “text view pane” on the screen.

1.2.2.3 Near or total blindness

People whose vision is so severe they have effectively no usable vision use a different assistive technology to read and edit ODF documents – a *screen reader*. This software uses synthetic speech (or *text-to-speech*, commonly abbreviated to *TTS*) to read the contents of the screen, application, and document to the user. Such software tracks the object or text the user is interacting with (sometimes called the *locus of focus*) and reads that object or text to the user (e.g. the letters the user is moving the text caret left-and-right between, or the lines of text the user is moving the caret up and down between, or the menu items the user is interacting with, or the items in the dialog boxes of the ODF application). Screen readers are sometimes combined with screen magnifiers, in which case magnification also tracks this locus of focus. Screen readers sometimes also support *refreshable Braille displays* in which case in addition to speech, information is rendered to the Braille display. If the user is both deaf and blind, Braille will be the exclusive medium of information from the ODF document and application.

1.2.2.4 Minor physical impairments

People with minor physical impairments have one of a variety of issues that prevent them from using a full keyboard easily, or prevent them from using the mouse. In order to support these users, ODF applications must first and foremost ensure that all document manipulation and other use of the ODF application can be accomplished easily without using a mouse. This is also a benefit to users who don't have a physical impairment, but otherwise cannot use a mouse (such as a blind user, who cannot see the mouse pointer on the screen and so is functionally unable to effectively use the mouse.

Where the physical impairment allows the user the use of only a single finger, or a *mouth stick* such that they can only make a single keystroke at a time, an operating system feature called *StickyKeys* would be employed that latches modifier keys such as Shift and Control in software, such that the next key pressed after the latched modified key comes through to the application as if the key and the modifier were pressed simultaneously (e.g. Shift then 's' to produce 'S'). Where the physical impairment results in hand tremors (such as Parkinson's disease), the user might hit keys accidentally en-route to pressing the key they intend to press. In such cases, another operating system feature called *SlowKeys* would be employed which rejects key presses unless they are held down for a specified amount of time. Where the physical impairment results in spasms that commonly generate a second, duplicate keystroke (a 'debounce') upon release of the key, an operating system feature called *FilterKeys* would be employed which rejects duplicate keystrokes unless a specified interval of time occurs between them.

1.2.2.5 Major physical impairments without speech recognition

People with major physical impairments are typically unable to control their limbs. They can perhaps shrug a shoulder, or sip and puff on a straw, or move their head or at least their eyes. But they are unable even to press a single key on a traditional keyboard. When these impairments also affect their speech (or if they simply choose to not use *speech recognition* software), such users use *on-screen keyboard* software that is connected either to a *switch device* that they can activate (a large button mounted to their wheelchair that they can press in some fashion, or a switch embedded within a straw that they can 'press' by sipping or puffing on that straw), or by head movement that is tracked by a camera following a reflective dot on their forehead or glasses, or by eye movement that is tracked by a camera following their pupil. This on-screen keyboard software uses a variety of techniques to translate these movements to choices of individual keystrokes, or to entire words, which are then entered into the system as if they came from the keyboard. This is done by displaing a software keyboard in a window on the screen, from which the user makes their choices. More sophisticated on-screen keyboard software can actually extract the menu structure of the ODF application, and the toolbar elements, and even the 'manipulable user interface elements' and place them on the software “keyboard” where the user can choose them.

Another, new type assistive technology used by people with major physical impairments is the dynamic text entry application, which couples a predictive text system with head, eye, or switch movement to rapidly enter the most common sequences of text based on a probability model for a given language (or a given language domain such as medical terminology). Users able to move nothing but their eyes are able to achieve more than 35 word-per-minute typing rates with such software.

1.2.2.6 Major physical impairments with speech recognition

People with major physical impairments who are able to speak clearly may use speech recognition software (also known as *speech to text software*, *automatic speech recognition* or *ASR*). [[[FIXME]]] MUST CHECK THE ABBREVIATION 'ASR' [[[FIXME]]]. This assistive technology not only translates the users speech to text (for entry into the ODF document), but also allows the user to control the ODF application via speech. For example, a user of this assistive technology might say the utterance “Menu File, Print” in order to choose the print item of the file menu. Or they might say “Select from 'Dear Mom' to 'please send money'” in order to select a range of text in the ODF document.

1.2.2.7Hearing impairments

[People with these impairment have difficulty hearing clearly, or perhaps cannot hear at all. A severe impairment may not be corrected by the use of a hearing aid. Users with this impairment use an operating system feature called *ShowSounds* and/or they use a different operating system feature called *SoundSentry*. ShowSounds is a system flag that indicates to applications that any speech and sounds made in a presentation should be captioned (e.g. text captions displayed in an mpeg video). SoundSentry causes the operating system to generate a visual warning when applications would otherwise made a warning tone.

1.2.2.8 Cognitive impairments

People with cognitive impairments have one (or more) of a very wide range of disabilities. These include a range of reading impairments (such as dyslexia), comprehension impairments, and composition impairments. Users with cognitive impairments use one of a variety of assistive technologies, and/or or application features, in order to more effectively use the computer. These include using a screen reader which reads document text via TTS and also highlights the words as they are spoken, on-line thesauruses and dictionaries with a special focus on homographs and homophones; and grammar checking tools. The most popular assistive technology software for users with cognitive impairments inserts itself into text-heavy applications like office suites, e-mail packages, and web browsers.

There are more severe cognitive impairments which to date haven't been addressed very well via either mainstream software or assistive technologies. Further research may improve the options available to these users.

1.3 Importance of the Accessibility of the ODF Application

Ensuring that the ODF application is accessible to people is critical for the following reasons:

1. People with disabilities together comprise one of the largest “minority” groups – nearly 20% of the worldwide population has some form of disability
2. In many countries, supporting people with disabilities in electronic and information technology is a legal requirement for sale or use of that technology in government
3. In many countries, providing an education to people with disabilities is a legal requirement, and many schools therefore will not purchase or use software that doesn't support people with disabilities
4. It is the right thing to do – morally, ethically

1.4 Importance of Ensuring Authors Encode Accessibility Information into Documents

The ODF v1.1 specification includes many optional elements that are critical for document content accessibility. Some examples are optional image text and descriptions, headings for tables, and logical navigation order for objects in a draw layer. It is vitally important that these optional elements be used by document authors to produce an accessible document. Therefore, it is important that ODF applications, at the bare minimum, allow authors to include these optional elements. Better still, the user interface presented to the author for doing this should be straightforward, simple, and easy to find and use. The idea ODF authoring/editing tool would have a feature or mode that audits ODF applications for accessibility, and flags the accessibility problems it finds. Just as virtually all ODF applications help authors to create documents without spelling errors and understandable grammar, a good ODF application will likewise help authors to create documents that are accessible to people with disabilities.

1.5 Putting the pieces together

Providing a document reading and editing experience that is accessible, efficient, and productive for people with disabilities requires multiple pieces of software working together. The tool that created the ODF document must have offered the author a way to encode the information needed for accessibility and assistive technologies (which the author must have used). The ODF document reader must expose ODF accessibility information to any assistive technologies running

on the operating system (ideally through a rich accessibility framework available on that operating system).

[see chart at http://www.w3.org/WAI/guid-tech.html to start]

2 ODF Application Accessibility

2.1 Keyboard Navigation or use without a mouse

Every feature, every function of the accessible ODF application must be available from the keyboard. Requiring the mouse as the sole means for accomplishing a task means that task won't be something many users with physical impairments can accomplish. Likewise blind users who cannot use the mouse because they cannot see where it is moving will be unable to accomplish that task.

However, it isn't enough to to simply support a sequence of keystrokes for accomplishing every task; that sequence of keystrokes should be efficient, and should follow conventions on that operating system for doing similar tasks (e.g. F10 for focusing the menu bar; ESC for canceling dialog boxes; TAB for moving between controls in a dialog box). See [[[FIXME]]] URL for the common keystroke sequences for Windows, UNIX, and Macintosh.

Another critical piece of keyboard accessibility support in the ODF application is compatibility with Operating System keyboard support features: things like StickKeys and FilterKeys. While it is unlikely there will be a compatibility problem, it is important to test the ODF application with all of these features to ensure that is the case. Code which examines the state of modifier keys upon reception of an event, instead of the modifier flags in the event itself, is a likely source of StickyKey compatibility problems.

2.2 Theme Support (including OS fonts & colors)

ODF applications must respect the color, contrast, and font choice information a user has made for their desktop. Users with a range of vision impairments (such as red-green color blindness) make these settings so that they can see user interfaces running on their desktop. If the desktop setting of font size is larger than standard (e.g. an 18 point font) the ODF application should ensure that no text in any menu, dialog box, or other user interface element is smaller than 18 point. Ideally the ODF application would further optionally scale document content, linked to the desktop font size setting, without requiring additional user interaction. This extends to every part of the ODF user interface, including things like print preview.

Where the underlying desktop offers additional settings for vision impairments – such as the UNIX desktop in GNOME which allows for custom large print, and high and low contrast icons – the accessible ODF application should behave in accordance with those options.

Where the underlying desktop fails the offer desktop settings for color, contrast, and font size – such as Macintosh – the ideally accessible ODF application will allow the user on that desktop to choose colors and fonts that work for them independent of the rest of the desktop.

2.3 Interoperability with Assistive Technologies

Blah blah blah

- Lots to say about desktop accessibility APIs...
- Note that this applies to help system as well

One of the most important things an ODF application must do to support accessibility is to be interoperable with assistive technologies. If, for example, the ODF application doesn't work with the screen read readers on a platform/operating system, then blind users won't be able to use that ODF application. If there is a basic level of interoperability between the ODF application and a screen reader, then blind users may be able to accomplish all of the reading and editing tasks, but not necessarily in a very efficient or productive manner.

The major operating systems are all either shipping, or developing, engineered accessibility frameworks specifically to facilitate interoperability between assistive technologies and software applications. The ODF application should support the accessibility framework, and fully implement its accessibility API, on the platform(s) it runs on. If there is more than one such framework on a given platform, support one that is well suited to complex office suites and that is well supported by assistive technologies on that platform.

2.3.1 Characteristics of Engineered Accessibility Frameworks

An engineered accessibility framework that can provide the rich support for interoperability with assistive technologies needed to yield a productive and efficient user interface for people with disabilities share a few common characteristics:

1. They provide a way to locate every user interface element – where it is both on the screen visually and where it is in the window hierarchy
2. They allow an application to describe every user interface element: what its role is (menu item, checkbox, etc.), and what states it is in (checked, selected, etc.)

3. They provide ways of conveying all of the details of the more complex user interface elements, including:
   a. the individual characters, font, font size, font style, text style, and character bounding rectangles of text
   b. text editing and selection information, including the location of the text caret, the ability to move the location of the text caret, and ability to programmatically cut/copy/paste text via the accessibility interface
   c. the ability to indicate hyperlinks with hypertext
   d. the minimum, maximum, and current value of objects like sliders and scroll bars that take on a range of values – and the ability to programmatically set the value
   e. whether an object presents one or more actions to the user (like a button that can be clicked, an object that exposes a popup menu with a variety of choices), and the ability to programmatically take one of those actions
   f. whether the object presents a table of information, and the ability to locate elements within the table by their row/column (and the ability to find the row/column of a given object within that table)

4. They provide a way of conveying at least basic relationships between user interface elements (such as a text label labeling an edit field, and elements being members of a group such as a collection of radio buttons)

5. They provide a way to provide events or signals, indicating asynchronously that any of a wide range of information changes to user interface elements, including:
   a. a user interface element appearing, disappearing, or moving
   b. a user interface element changing stated (e.g. from selected to unselected)
   c. text being added/replaced/deleted
   d. text changing font, font style, etc.
   e. the text caret moving, or the text selection changing
   f. the value of an object that takes on values changing

6. They are extensible, and as new user interface elements are developed, they can be extended to communicate any new and unique aspects of such new elements

7. They can be implemented independent of any particular user interface toolkit or library used. Thus an ODF application isn't restricted to using (for example) the GTK+ user interface library and thus be automatically ATK accessibility toolkit in order to be accessible in a UNIX environment; it can implement ATK or AT-SPI directly on that platform.

2.3.2 Recommended Engineered Accessibility Frameworks

The following accessibility frameworks are recommended for ODF applications to use on their respective platforms:

- On UNIX systems, use the GNOME Accessibility framework. This can be done via one of several specific user interface toolkits, including GTK+, UNO, XUL, and Java/Swing. Or it can be done by implementing support for ATK or the Java Accessibility API directly, or by AT-SPI directly. In any case, it is highly likely that either ATK or AT-SPI support will need to be implemented for the editing/content portion of the ODF application. This is well supported by UNIX assistive technologies such as the Orca, LSR, and Gnopernicus screen reader/magnifiers, and the GNOME On-screen Keyboard.
- On the Java platform, use the Java Accessibility API. This can be done by using Java/Swing directly, or by implementing support for the Java Accessibility API directly. This is well supported on UNIX systems via the GNOME Accessibility framework (which bridges the Java Accessibility API out of the Java Virtual Machine), and is unfortunately only poorly supported on Microsoft Windows by assistive technologies via the Java Access Bridge for Windows.
- On the Macintosh (OS X v10.4 or later), use the Apple Accessibility API. This is supported by the built-in VoiceOver screen reader, and the built-in magnifier.
- For web-based ODF applications, use the WAI-ARIA interface, which today is supported by the Firefox 2.0 web browser on Microsoft Windows and several commercial assistive technology products including the JAWS and WindowEyes screen readers. Support for WAI-ARIA is anticipated on UNIX with the release of Firefox 3.0.

2.3.3 Supporting an Accessibility Framework

Whether the ODF application uses an existing user interface toolkit which supports an accessibility framework, or implements all of the framework support itself, it is important that the ODF application accurately expose all of the information about all of the user interface elements that the accessibility framework can communicate. And it is especially important that the ODF application fire events/signals whenever the user interface elements change the state, gain/lose focus, or their contents (e.g. text) changes. This is critical because many assistive technologies are event driven – reacting to text appearing by speaking it to the blind user, reacting to the caret moving within the text by reading the letters being moved through to the blind user, and moving the magnified view to track the caret for the low-vision user.

The ODF application development and test team should test to ensure that the accessibility framework is properly implemented – both by using test tools, and also by using the ODF application through the assistive technologies available for that platform. This is ideally done by people with disabilities who are experienced with using the assistive technology in question. However, even just turning off the monitor for a day and using the ODF application via a screen reader (for example) is extremely helpful.

2.3.4 Dealing with the Absence of an Accessibility Frameworks

Today not every platform provides an accessibility framework that is sufficient to convey the richness of an ODF application, and further which is supported by the assistive technologies on that platform. Unfortunately in those cases the only option may be to form a business relationship with one or more assistive technology vendors who may then either support some non-OS-defined accessibility framework that the ODF app and AT can agree on, or extend the reverse engineering techniques they otherwise use to provide access to that platform, or some combination of the two.

This is the situation today in Microsoft Windows, where the Microsoft Active Accessibility API is not adequate to support interoperability between something as rich as an office suite and an assistive technology. This situation is expected to change someone in the next few years, with the release of Microsoft Vista and the adoption of UI Automation, Microsoft's second generation accessibility framework.

2.4 Special Issues for Web-based ODF Applications

For web-based ODF applications, accessibility poses special challenges. The accessibility of the ODF application is significantly dependent upon the web client being used. Web-based ODF applications should utilize the W3C WAI-ARIA specification (currently in draft form) for conveying all dynamic/interactive portions of their interface. Use of WAI-ARIA - through a browser that supports WAI-ARIA such as Firefox – provides support for assistive technologies by conveying the content, meaning, context, and updates/changes of the user interface.

Further, the web-based ODF application must respect user-settable CSS settings in order to support color, contrast, and font settings the user indicated either on their desktop, in their web client, or via their own custom CSS.

As of this writing, the best choice of an accessible web client for dynamic web applications on a desktop computer is Firefox on Windows. Work to support exposure of WAI-ARIA accessibility API information via Firefox on UNIX and

Macintosh systems is underway. The UNIX work is expected to finish in Firefox 3.0 by mid-2007.

2.5 ODF Help System Accessibility

Help systems are often separate from the application they are providing help for. From the end-user point of view, help is a feature of the ODF application. An ODF application isn't fully accessible unless the associated help system is likewise accessible. All of the accessibility provisions described above therefore also apply to the help system.

3 Document Content Accessibility Basics

Access to content for people with disabilities has a number of aspects. These are discussed here.

3.1 Document Navigation or Structure

Well written documents generally have a few meaningful characteristics. These are a meaningful title, a brief summary of what the document contains, and regular structure. The title should tell us if we are interested in the contents, the summary tells us about the document body, and the structure enables users to find their way around the document. Access to the parts of a document that provide these clues is key to navigation. A title is visible title text marked in XML as text:h. It is important that implementations support the user in generating appropriate XML. Visually bold centered text is not the same as a title as far as XML is concerned.

3.1.1 Text document structure

Styles, structure and visual presentation. Named styles (heading 1, Default, List indent etc) are available to the user and provide structure in the saved XML file on disk. Additionally each of these named styles may be formatted for a given visual presentation. It is important to realize that the appropriate use of styles helps greatly with accessibility. Please use named styles to generate your document, since this provides the foundation for the document structure. It also helps when generating a table of contents and cross-references. Perhaps the biggest aid is that all content with a given style can be given a difference appearance with a single change, instead of changing hundreds of individual pieces of content!

If I create two pieces of content which are visually identical, one using named styles and the other using styling attributes, it is important to realize that the XML produced will be different for each.

The structure, for an ODF document (for a text document), refers to the use of named styles. The automated processing of style information can be used to generate tables of content, cross references and more importantly, present the overall structure of the document for the reader. When saved to it's default form, an ODF document provides a structure based on those styles which allows

navigation and access for other tools, using the Extensible Markup Language (XML) semantic markup.

Hence the structure is important for more than one reason. When using styles a user needs to make a visually pleasing presentation, ensure that the XML produced reflects the structure such that access technologies can make use of that information. Other XML tools can be used to abstract a quick overview if needed, by selecting just those styles which are key to the document structure. If a Text style is used, with a larger font and centered layout, it gives the appearance of being a heading, but that semantic is lost in the exported XML. Implementers need to assist the user in using named styles as apposed to modified font sizes and other presentational aspects

There are three possible ways to obtain entries in a table of contents. The headings (heading 1 etc), Table of content index marks and named paragraph styles. The table of contents may be used as a document overview, so it is worth while using one to aid accessibility and help readers navigate the document.

3.1.2 Presentation document structure

For a slide presentation similar logic may be used. The title of each slide presents a variation of a table of contents, providing the person using access technology with an overview of the presentation content. If the author provides meaningful titles, an outline of the presentation is available, even if the slide body content is inaccessible. Beyond this, it is largely down to the author to help a user comprehend the presentation structure.

Within a single slide, ODF has the idea of a logical navigation order (see ODF 1.1, section 9.1.4). This enables the author to specify how the slide should be *read.* If the implementor helps the author specify this navigation order then the end user can readily navigate between the objects on a slide in the order that the author intended. For example, a keyboard user may use the TAB key to skip between the images on a slide, and access technology can then present information about the image based on the textual alternative description. Only in this way can non visual access to slides be made accessible.

3.1.3 Spreadsheet document structure

Unless a spreadsheet has multiple sheets there is little structure to inform the reader about the sheet. What can provide an overview is a good heading to the sheet. If the implementor makes it easy for the author to add an informative title to the sheet this helps with access. Other than by using row and column headings a regular matrix is difficult to navigate. Row and column headings

provide an anchor for the reader when navigating them. If data is accessed serially, as is the case with someone accessing it via a text to speech system or a braille system, then it is easy to forget whether this is the debit or credit column (or whatever the heading is). The provision of meaningful headings is vital in this case. The implementor can help by making it easy for the user to add row and column headings and ensuring that these are correctly marked up in the saved XML.

3.1.4 List structure

Lists provide an opportunity to confuse structure and presentation. Nested lists are visually indented from the parent list (perhaps using an increase indent control). This is not the correct way to produce well structured lists. Users are advised not to simply increase the indent level to nest list items.

Implementors should provide an appropriate nested structure for nested lists and help users create lists that way.

3.1.5 Page breaks.

When sighted readers talk about printed matter, a common frame of reference is to use page 34, third paragraph down. Unless a non-visual reader can access the pages using this frame of reference there is clear exclusion. Implementors should provide notification of page breaks on export.

The visual page breaks a sighted user can see (in print layout mode) should be made available to access technology.

3.1.6 Table navigation.

This is especially important for tables which span multiple pages. For non-visual users of ODF applications, particularly text documents using tables, it is necessary to indicate clearly which cells contain row and column headings. Implementers can help by prompting users to identify headers and marking them syntactically within table:header-row. This allows access technology to inform the user what the column header is for the cell having current focus. Again, users should ensure that those cells are structurally marked (named styles) rather than visually marked (emphasis etc). Implementers should ensure that table header content is available to access technology no matter what API is in use. The rationale here is to ensure that users have access to header information even when a number of pages into a large table. It will be helpful to sighted users if table headers are repeated on each page.

3.1.6.1 Tables within presentations.

ODF does not support structural tables within presentations. Implementers can help by using embedded spreadsheet tables, which do have a structure. This is another case where visual and structural information can differ. Embedded spreadsheet tables have a fully navigable structure.

3.2 Provision of alternative text versions of document content.

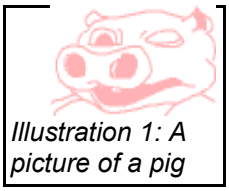


Illustration 1: A picture of a pig

When a non-visual user reads your document he or she will not have access to any diagrams or images included within it. Attached to this paragraph is an image. What can you find out about it? For a reader using access technology, the provision of an alternative form is essential if he or she is to gain access to the information, which is what the author generally wants. The implementor can help by making it easy for the author to add an alternative (generally a textual alternative) to the image. If the manner in which they add such an alternative is consistent across the office suite, then it becomes second nature to add this content. If they have to select different drop down menus, with different fly-out options in each application, then interest is soon lost. Choose a method which is common across applications and help the author add content easily. If this method works with line drawings, embedded images, audio clips etc, then the author gains confidence in the application and can help the reader gain access to the information provided by the author.

The image may be described in one of two ways. A simple decorative image can be described using the svg:title element. For a more complex image (or diagram etc.) there is the svg:description element (equivalent to the html longdesc) which provides a way of entering a more complete alternative to the visual object.

Next there is the draw:caption element. Implementors can help by encouraging the addition of captions (often a few words providing a label for the object which are visually associated with the object). Captions are associated with the description by means of the draw:caption-id attribute. This must be added to link the object and its description.

Implementors must provide the means to add these descriptions in a consistent manner across all office applications.

3.2.1 Imagemaps

It is possible to assign areas of an image as being active with a hyperlink to another uri. This is commonly known as an imagemap. As with any image, a brief description should be provided using the svg:title element, or a longer description using svg:description if a longer text alternative is needed. These text descriptions should provide information about the content at the end of the link. Implementers can help the user to enter this by providing a standard dialog letting users enter text for the alternative descriptive text.

When a reader with a hearing impairment receives one of your documents will he or she hear what you inserted? Attached to this paragraph is an audio object. What do you know about it? What level will it play at? How can a reader with a hearing impairment adjust the playback volume? If a reader has insufficient auditory sensitivity to access the clip, how can the implementor help the author provide an alternative rendition? The most general method is to provide a textual alternative. How easy is it for an author to add that? Must he or she learn a new technique? Help the reader by helping the author.

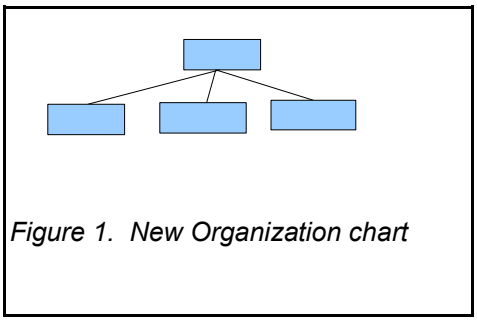


Figure 1. New Organization chart

When you enter your new organization chart as a part of a report, do you rely on people having full and easy visual access to it? If it were you, looking for your new position I guess it would be important. What if you couldn't see it or your eyesight isn't sufficient to make out the detail? Does that mean it's of no interest to you? Of course not. this kind of consideration matters if the information held in the object is important to the understanding of the document. How can the implementor help the reader? Again ensure a consistent way in which the author can add a simple textual alternative. To the reader, the graphic is a variant form of an image, to the author, the alternative text is another way of presenting the information. Help them by making it easy to provide that alternative.

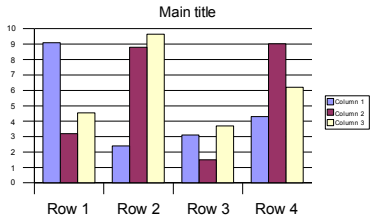


Illustration 2: Sales Graph, 1998 to 1999

Business reports often include graphs. I've included a dummy sales graph linked to this paragraph. It may be important to your report, it could be a crucial element of it. If the reader can't get at the information then the authors efforts are of little use. The reader needs to be aware that it is there and to have access to a format that they can access. For a complex graph the textual alternative could be equally complex. Can you provide facilities which enable alternative text from a few words to a 100 word description? Does your input area flex with use? Try using it yourself to describe a couple of graphics you have seen in text documents.

3.2.2 Summary

Each of these (and there will be more) shows how we individually access information. Many of us scan information visually. Others don't or can't. If you provide equal access to all readers an author can feel confident in reaching a maximum possible audience. If an author writes without consideration for the audience they may totally miss a key piece of information. If an implementor makes it easy for an author to consider and meet the needs of the audience then the objective of transmitting information is achieved. A subsequent section of the document may depend on them understanding that the 1999 sales figures shown in a graph have repercussions which the remainder of the document then explain. If a reader has totally missed that graph (for whatever reason), then the document may as well be thrown away. If you know all the recipients of your document then consider them all. If you are writing for a general audience then this appendix will provide guidance on how you can make it easier for them to access the information you are passing on.

3.2.3 Why do anything about it?

Consider receiving a document where one third of it is blacked out, or blank. Even if the remainder makes sense, you are left wondering what you have missed. Censorship may be the cause, or a careless writer. If you are unable to access visual, auditory and textual content in a document then that is exactly the situation you are in when you receive a document prepared without consideration for accessibility.

3.2.4 What to do about it?

Of course you can include any of these objects in your documents. Just make sure that the objects have sufficient information for access using different access methods. I use sight and sound. Norman uses sound and a braille display (tactile), Kate uses sight only. If you want to cater for a wider audience, consider what you are creating and the form you are using to create it. We can talk about different modalities, ways of accessing information. Sight, sound and touch (tactile access) are the most common.

It may be useful to know that assistive technology can provide a user whose preference is audio with an auditory feed by capturing the text under the cursor and synthesizing the text which the user listens to. As you can appreciate, such access is effectively serial. There is no fast way to navigate a document, what is read out to you is what is under the cursor! If an implementor tries this with a spreadsheet then the task becomes somewhat more complex.

A basic item is the font and font size that you select. Personal preferences vary from tiny 6 point text through to big and bold 16 point body text. For clarity and ease of access by a majority, try and work with font sizes of at least 12 point, and choose a font without serifs (those fine lines that decorate the top and bottom of verticals), commonly identified as sans-serif. Good strong clear fonts are Arial, Helvetica and Verdana. Research shows these to be clearer than most for users with less than perfect sight. Choose one of these from the fonts you have available and offer this as a default. Don't make presumptions about how your readers will access your documents!

Arial, 12 point.

Verdana, 14 point.

Arial, 10 point.

3.2.5 A few examples

For images

If you want a frivolous graphic simply to brighten up a document, let the user with no visual access to the information know that. Provide an alternative text label that describes the graphic as a decorative one (with no informational content). Another useful attribute of a picture is a name or title. Again this can be useful in certain circumstances, so make it easy for an author to add a title (if you name it as a caption, make that nomenclature consistent across applications).

For drawings

The information contained in a (line) drawing is often quite different from an image.

To access the information contained in the drawing without using your eyes is an identical problem. Provide the author with the same access techniques. Is it possible to provide textual alternatives for a group of lines or objects? Must each line be described - which is rather tedious. How can the implementation help a user to provide a flexible means of providing an alternative?

For graphs and Tables

Treat these as for drawings. Sometimes people who work with numbers naturally translate information into tables and associated or derived graphs. Is that the best way to present the information? You can decide. Tabular data is not easy to access serially, as is done by a tactile user. Navigating tabular data the current position needs to be recalled (row and column), headings for both need to be remembered. It is not easy. It is really helpful to provide headings for rows and columns so that they can be accessed when needed, just as a sighted user glances at the top of a column a tactile user needs to use keyboard strokes to navigate to the heading, then back to their old position. Graphs should be treated just as images, even if embedded into a table or spreadsheet. If the information is key to understanding, make it easy to provide a full textual description.

For Audio objects

Audio is not a natural companion to a text document. If you have added audio to a presentation then let's assume it serves a valid purpose. To see how comprehensive the document is without the audio, turn off your speakers and see if the information is still available. What is missing? How can an alternative be provided for the reader? Is a textual description needed? Is the audio for effect only? Let the reader know about the audio. If they are left without such information, then concerns and doubts will arise, wondering what they have missed. Add a few words indicating what the audio contains.

3.2.6 Summary

For each change of media, think of potential readers who may not be able to access it. If your information is important, provide an alternative for that reader.

3.3 Special Considerations for Subtables

It is suggested that a User Agent not use or allow the use of table:is-subtable. This is because the ODF specification for subtables requires the use of a special cell addressing scheme. Cell addresses are an important structural aid for users who are blind and thus not able to visually see the spacial location of a cell within its table. Since the subtable addressing scheme is different than the norm, it use can easily be disorienting as described below. The same visual effect can be obtained using table:number-rows-spanned or table:number-rows-spanned and the use of these tags will result in cell addressing which is consistent with the surrounding table.

A1	***B1***	***C1***
A2	.B2.A1	.B2.B1
	.B2.A2	

Sample 1

Sample 1 is an example of the problem that occurs when using table:is-subtable. In this example each cell contains its cell address. The semantics conveyed by cell addresses are especially important to those who are using an assistive technology (AT) such as a person who is blind. An AT will expose information about the cell which currently has keyboard focus. Without significant extra effort a user who is blind is not able to “see the entire table. An important piece of structural information is the cell name, for example A1, B2, etc. The use of table:is-subtable results in a modified addressing scheme and that addressing scheme conveys important structural information to a blind user. For example, in sample 1 there is a B2 (.B2.B1) under an C1. This will be disorienting.

An example of why an office application should not use table:is-subtable is a table with row and column headers. In Sample 1, if the gray cells, i.e. cells A1, B1, C1, and A2 are the row and column headers of the remaining cells, then the use of table:is-subtable results in a cell addressing scheme that is disjoint from the addressing scheme of the row and column headers and the semantic relationship is broken. The visual equivalent of Sample 1 is Sample 2, just below which will result when using table:number-rows-spanned or table:number-rows-spanned. In Sample 2, cell C1 is the header of cell C2. But in Sample 1 cell C1 is the header of a B2 cell.

A1	***B1***	***C1***
A2	**B2**	**C2**
	B3	

Sample 2

4 Converting between ODF and other Document Formats

Accessibility issues surrounding document format conversion are often overlooked. This section covers issues that an office application must conserve to preserve accessibility when converting between ODF documents and other Office documents such as HTML, DAISY, or MS Office.

4.1 Preservation of Alternative Text

Alternative text is used to provide alternatives to non-text objects such as drawing or images. Throughout the ODF specification, **<svg:title>** and **<svg:description>** are use to represent the short accessible name and long description of these document elements. Long descriptions are not new to HTML however they are new to office documents formats and consequently long descriptions may have no mapping.

Alternative text is used to describe a graphic or drawing object. Users who are blind prefer to hear a short description for these document elements when navigating a document with the screen reader as it improves productivity over using a long description. The ODF Accessibility Subcommittee introduced long descriptions for drawing objects to provide more information. This is particularly important for complex drawings formed from a group of smaller shapes.

When converting to and from ODF it is essential that alternative text is preserved.

4.1.1 <svg:title>

The <svg:title> attribute represents the short accessible name of the non text drawing object.

Need to ensure that the double quotes come through in the submitted document. Not correct in Open Office 2.0

ODF 1.1 Element	**HTML element using alt=""**	*HTML element using title=""*
<draw:rect> <draw:line> <draw:polyline> <draw:polygon> <draw:regular-polygon>	<IMG alt="">	

ODF 1.1 Element	**HTML element using alt=**	*HTML element using title=*
<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:a>	<a>	
<draw:layer>		
<draw:image>	<IMG alt=>	
draw:area-rectangle	<AREA Shape="rect" alt=>	
draw:area-circle	<AREA Shape="circle" alt=>	
draw:area-polygon	<AREA Shape="poly" alt=>	

4.1.2 <svg:description>

The <svg:description> element represents the long description of a drawing object.

ODF 1.1 Element	*HTML element using alt=‿*	*HTML element using title=‿*	*HML element using longdec*
<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>		<img title=‿>	
<text:a>		<img title=‿>	
<draw:layer>		<img title=‿>	
<draw:image>		<img title=>	<img longdesc=””>

Longdesc is a special case situation. It should only be addressed when importing from an HTML document. In the case where longdesc refers to a separate file, the file may contain a significant amount of HTML content. The user agent would need to decide it if would prefer to consume the contents of the file and store it in a test string. Typically, the title text would suffice. Very few web page authors make use of longdesc as it requires as significant amount of work to create the additional file. Use of the title attribute to store significant amounts of text would be easier to do.

4.2 Preservation of Document Structure Hierarchy and Landmarks

Preservation of document structure is critical for accessibility. Structural information is used to give the person with a disability context of where they are within a document. The following structural information must be preserved when converting between document formats:

The bullets or list markers are corrupted in OOo.

- Heading elements including their level (perhaps depth? Or hierarchical level)
- List structural elements
- Footer elements
- Speaker note elements
- Tables
- Page breaks and page numbering

Preservation of this information must be done by ensuring corresponding tags in the content model exist and are mapped during format conversion. Use of styling is inadequate. Content, and not styling, is mapped to platform accessibility application interfaces (APIs) to convey structure to assistive technologies.

4.2.1 Preservation of Heading Structure

ODF supports headings and heading levels. It is important that headings and their levels be preserved to preserve structural semantics.

As an example, we will cover the conversion of ODF heading and heading levels to HTML. Any <text:h> element which does not provide a level, in the form of the text:outline-level> attribute is considered a level one header. HTML includes elements pertaining to H1, H2, H3, H4, H5, H6. Office applications should consider limiting their heading levels to the document formats the support exporting to.

Ignoring styling, the following header content:

```
<text:h text:style-name="Heading_20_1" text:outline-level="1">Heading 1</text:h>
<text:p text:style-name="Text_20_body">Sample text</text:p>
<text:h text:style-name="Heading_20_2" text:outline-level="2">Heading 1.1</text:h>
<text:p text:style-name="Text_20_body">More sample text</text:p>
<text:h text:style-name="Heading_20_3" text:outline-level="3">Heading 1.1.1</text:h>
<text:h text:style-name="Heading_20_1" text:outline-level="1">Heading 2</text:h>
```

converts to the following HTML:

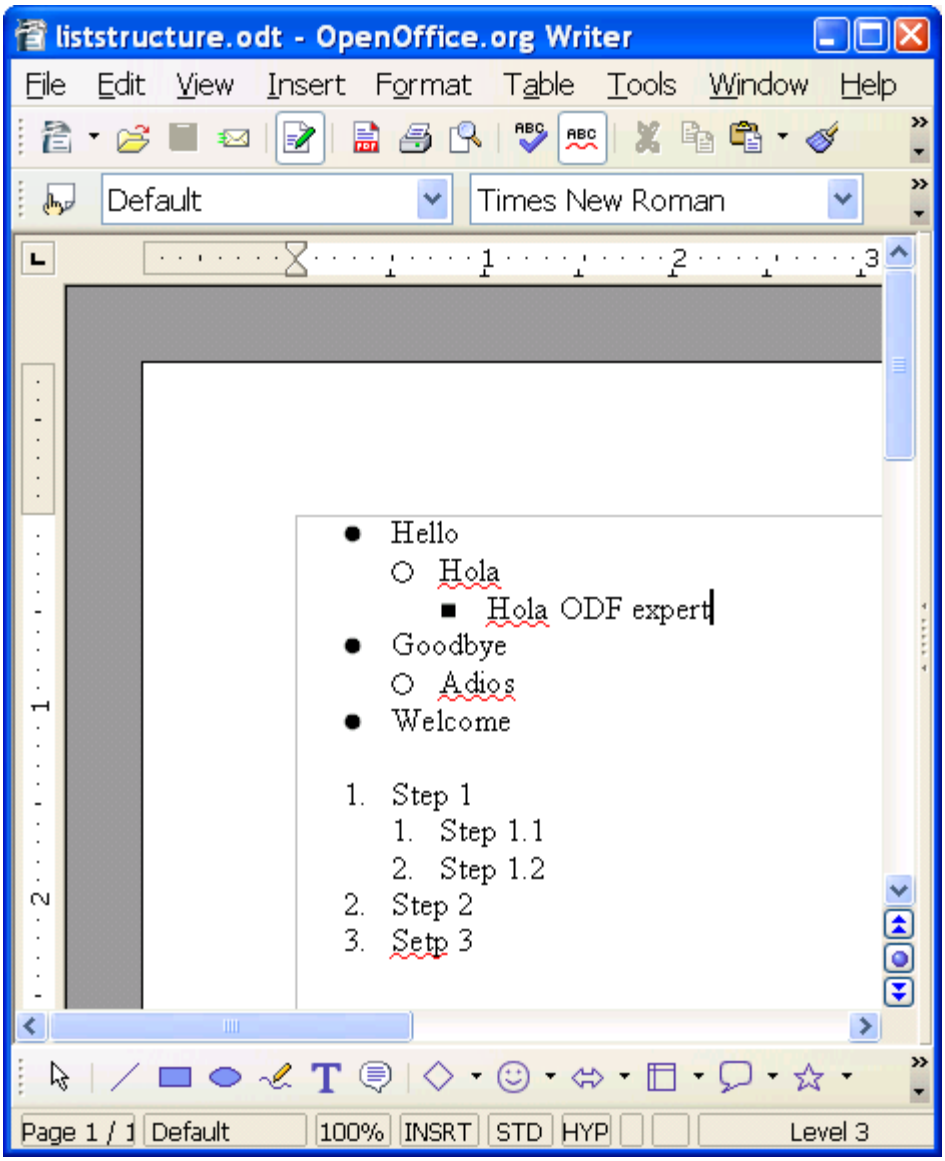
```
<H1 CLASS="western">Heading 1</H1>
<P>Sample text</P>
<H2 CLASS="western">Heading 1.1</H2>
<P>More sample text</P>
<H3 CLASS="western">Heading 1.1.1</H3>
<H1 CLASS="western">Heading 2</H1>

Where did the class attribute come from?
```

Although levels in ODF headers are defined by attributes the conversion is straight forward given the predefined heading levels in HTML. By preserving the heading levels in the content an office application can map the information to standard accessibility API's to be rendered in an alternative form at such as a screen reader would to using speech.

4.2.2 Preservation of list Structure

When converting to and from ODF it is important that the lists structural hierarchy is preserved. Assistive technology can use this information to convey contextual list depth information. When an office application uses styling to perform the indentation rather than list structure accessibility is lost.

As an example the following shows a conversion from ODF list structure to HTML list structure.



Here is the ODF Content:

```
<text:list text:style-name="L1">
    <text:list-item>
      <text:p text:style-name="P1">Hello</text:p>
       <text:list>
         <text:list-item>
            <text:p text:style-name="P1">Hola</text:p>
             <text:list>
               <text:list-item>
                    <text:p text:style-name="P1">Hola ODF expert</text:p>
              </text:list-item>
```

```
            </text:list>
          </text:list-item>
        </text:list>
    </text:list-item>
     <text:list-item>
        <text:p text:style-name="P1">Goodbye</text:p>
          <text:list>
           <text:list-item>
              <text:p text:style-name="P1">Adios</text:p>
           </text:list-item>
         </text:list>
     </text:list-item>
     <text:list-item>
        <text:p text:style-name="P1"> Welcome </text:p>
     </text:list-item>
</text:list>

<text:list text:style-name="L2">
    <text:list-item>
       <text:p text:style-name="P2">Step 1</text:p>
       <text:list>
          <text:list-item>
              <text:p text:style-name="P2">Step 1.1</text:p>
          </text:list-item>
          <text:list-item>
              <text:p text:style-name="P2">Step 1.2</text:p>
          </text:list-item>
       </text:list>
    </text:list-item>
    <text:list-item>
        <text:p text:style-name="P2">Step 2</text:p>
    </text:list-item>
    <text:list-item>
        <text:p text:style-name="P2">Setp 3</text:p>
     </text:list-item>
</text:list>
```

Converted HTML content (without styling):

```
    <UL>
```

```
      <LI>Hello
         <UL>
             <LI>Hola
                 <UL>
                    <LI>Hola ODF expert</LI>
                 </UL>
              </LI>
         </UL>
      </LI>
      <LI>Goodbye
         <UL>
            <LI>Adios</LI>
         </UL>
      </LI>
      <LI>Welcome</LI>
  </UL>
  <OL>
      <LI>Step 1
         <OL>
              <LI>Step 1.1</LI>
              <LI>Step 1.2</LI>>
         </OL>
      </LI>
      <LI>Step 2</LI>
      <LI>Setp 3</LI>
  </OL>
```

Note the font change again.

To properly preserve content structure, each embedded list must be enclosed within the enclosing list item as it has been done in ODF. In this example an assistive technology can accurately convey the property list depth information as well as the number of siblings at a given level.

4.2.3 Preservation of Page Header and Footer structure

Preservation of page header and footer information in a page, when supported between file formats, is essential to preserve structural context and to allow the user to navigate to these sections of a page. In ODF, this information is stored in the styles.xml file within the master-styles section as shown in this example:

```
<office:master-styles>
  <style:master-page style:name="Standard" style:page-layout-name="pm1">
    <style:header>
```

```
      <text:p text:style-name="Header">This is a header</text:p>
    </style:header>
    <style:footer>
      <text:p text:style-name="Footer">This is a footer</text:p>
    </style:footer>
  </style:master-page>
</office:master-styles>
```

Neither HTML nor XHTML support a page header or footer construct. However, a new role attribute has been created by the HTML working group which provides standard roles for standard web page landmarks as part of its new XHTML Role Attribute module. The standard landmark of contentinfo may be used to identify the role header and footer for a page. Currently, Open Office drops the header and footer information when converting to HTML.

4.2.4 Preservation of Speaker Note elements

When converting from one presentation format to another, the office application must preserve a clear delineation of the speaker note. If speaker note sections are only identified using styling assistive technology may not be able to identify to the user the contents of the speaker notes. In short, speaker notes should be considered a structural landmark. (Should this be metadata perhaps? I don't understand structural landmark)

The following example denotes an ODF speaker notes section using the presentation:notes element.

```
<presentation:notes draw:style-name="dp2">
 <office:forms form:automatic-focus="false" form:apply-design-mode="false" />
 <draw:page-thumbnail draw:style-name="gr1" draw:layer="layout" svg:width="12.768cm" svg:height="9.576cm" svg:x="4.411cm" svg:y="2.794cm" draw:page-number="1" presentation:class="page" />
 <draw:frame presentation:style-name="pr3" draw:text-style-name="P3" draw:layer="layout" svg:width="15.021cm" svg:height="10.63cm" svg:x="3.292cm" svg:y="13.299cm" presentation:class="notes">
 <draw:text-box>
 <text:p text:style-name="P3">This December, 1991 Byte article was written by Rich Schwerdtfeger with side bars by Joseph Lazzaro</text:p>
 </draw:text-box>
 </draw:frame>
</presentation:notes>
```

When converting to HTML, there is no element which would denote a speaker note. However, a new role attribute has been created by the HTML working group which provides standard roles for standard web page landmarks as part of

its new XHTML Role Attribute module. The standard landmark of note may be used to identify the role header and footer for a page. Rendering of the speaker note could be performed through the use of script which changes the visibility of the speaker note section through the use of a Cascading Style Sheet. Currently, Open Office the speaker note information when converting to HTML.

4.2.5 Preservation of Table structure

Preservation of table structure is essential for a disabled user to navigate a table. A user who is blind cannot perceive where they are in the table without preserving context. The following must be preserved when transforming one document format to the other:

The bullet (list markers) have not translated properly.

- Row Header information
- Column header information (if supported)
- Row demarcation elements
- Cell demarcation elements
- Captions (if supported) – may need to move this to alternative text section
- Spreadsheet Equations must have cell references consistent with table structure

ODF 1.1 does not support native tables in presentations. Users importing non-ODF slides that contain tables need access to the table structure via their assistive technology. Therefore tables imported into an ODF application from another file format must have their structure preserved, and when saved as ODF should be saved as embedded spreadsheets. The ODF Technical Committee is reviewing adding native table support in a future version of ODF.

4.2.6 Preservation of Page Breaks

A blind user or mobility impaired user may wish to navigate by page as this reduces the number of keystrokes required to navigate a document. In fact, ODF 1.1 has introduced a soft page break. When combined with hard page breaks (injected by the user and not through automatic pagination) the two items in combination allow for page-based navigation in a DAISY digital talking book conversion. DAISY readers are used by blind and partially sighted people.

4.3 Maintaining the accessibility of Form Elements

The following accessibility features of form elements must be maintained

- label associations within form controls
- form control titles

ODF borrows these concepts from HTML forms and although ODF claims to use XForms it does not use XForms controls. In XForms, label would be a child of the form control it labels. In ODF, a label references a form control, like HTML, by using the form:for attribute and using the form:id of the control it is labeling. The following example shows the use of label and title as applied to a date field.

```
<form:textarea form:name="TextBox" form:control-implementation="ooo:com.sun.star.form.component.TextField" form:id="control1" form:convert-empty-to-null="true" form:title=‿Enter your Date of Birth‿>
 <form:properties>
  <form:property form:property-name="DefaultControl" office:value-type="string" office:string-value="com.sun.star.form.control.TextField" />
 <form:property form:property-name="MultiLine" office:value-type="boolean" office:boolean-value="true" />
 </form:properties>
</form:textarea>
 <form:fixed-text form:name="LabelField1" form:control-implementation="ooo:com.sun.star.form.component.FixedText" form:id="control2" form:label="Date" form:for="control1" />
```

When exporting to html the label and it association to the form field, as well as the title, should be preserved as follows (note this is without full styling):

```
<FORM NAME="Standard">
  <LABEL for="control1">Date</LABEL>
  <P STYLE="margin-bottom: 0in">
     <TEXTAREA NAME="TextBox" ROWS=3 COLS=29 WRAP=SOFT STYLE="float: left; width: 2.48in; height: 0.65in id=control1" title‿Enter your Date of Birth‿></TEXTAREA>
     <BR>
  </P>
</FORM>
```

Note: OpenOffice does not yet support this feature when converting to html.

4.4 Maintaining Association Captions

ODF allows authors to create a caption for drawing objects. Structurally, it is not clear that the caption is associated with the drawing object. To address the problem a <draw:caption-id> was created to establish a relation between the

drawing object and the text caption describing the object. The id refers to the XML:id of the prose describing the drawing object.

When converting to other document formats it is important to maintain the relationship between the caption and the corresponding drawing object as long a a mechanism is provided.

When converting to HTML, drawings are converted to images. Although HTML, natively, does not provide a caption relationship between the image and the corresponding prose it does provide another association called the long description or longdesc. In this code example a longdesc links to a paragraph containing a unique id defining the prose text. Additionally, an HTML anchor containing Figure 1.0, from the start of the text, links backward to the named anchor representing the start of the image. By maintaining the relationship between the image and its caption it makes it easy for the user to find the caption text for a given drawing.

```
<a id="fig1" name="fig1" />
<img width="640" height="512" alt="Accessibility infomation mapped to a
DOM element in the DOM" src="accessibleelement" longdesc="#desc_fig1"/>
<br />
Figure 1.0 Accessibility Interoperability at a DOM Node without
JavaScript</p>
<p id="desc_fig1">Long description of the figure....
<a href="#fig1">Figure 1.0</a>
```

It is important to note that if other document formats do not have provision for addressing captioning then exporting ODF to that document format will result in a reduction in accessibility.

4.5 Preservation of MathML accessibility information

MathML is the accepted standard for representing mathematical notation in XML. Accessibility is an important goal of MathML since its inception. It can be used by synthetic speech applications or translated to Braille math codes for use with an embosser to print hard copy Braille or on a refreshable Braille display connected to a computer. MathML supports navigation within a mathematical expression. MathML also provides support for synchronized highlighting of what is spoken for the math, which is an important accessibility consideration both for users with low vision as well as people with print related learning disabilities such as dyslexia. For these reasons, MathML is being incorporated into other accessibility standards, such as DAISY and other ebook and document standards. MathML is also an accepted standard for interoperability between computer applications. Two of the many applications that can accept and/or produce MathML are Microsoft Office 2007 and Mathematica.

MathML can be exported to XHTML for use by Firefox or Internet Explorer when used with the (free) MathPlayer plug-in, which interfaces with screen readers to make the math accessible. MathML is supported in HTML by Internet Explorer and in an experimental version of Firefox, but in general, MathML support in HTML is problematic if cross-platform compatibility is desired. One industry approach to cross-platform compatibility has been to use the object tag in HTML. However, Internet Explorer's implementation of object tag does not follow the W3C guidelines and its rendering results are poor, so this approach is therefore to be discouraged.

An img tag with alt text does provide cross platform compatibility, but at a cost of sacrificing all but the most rudimentary level of accessibility to math. Math images cannot be enlarged well for large print, nor can their colors be changed for those with color disabilities. Math equations supplied in image formats cannot be navigated (other than the description supplied in the alt text), cannot be translated to Braille, nor can synchronized highlighting be supported. It is also important to note that generating good alt text for math is not simple because it must be unambiguous. For example, "1 over n plus 2" can mean "1/n+2" or "1/(n+2)". Furthermore, certain categories of readers may require enhanced verbosity while other may find this more confusing. Although images with alt text might seem like an accessible alternative, their use should be discouraged for the reasons listed above, and MathML should instead be used whenever possible.

4.6 Preservation of Synchronized Media (animations) SMIL

Animations in ODF applications are achieved using the Synchronized Media Integration Language or SMIL which does support accessibility. SMIL does support accessibility by allowing for synchronized text with audio for audio formats. ODF only uses SMIL to synchronize animations such as fly ins. For this, no additional work is required as long as when content is rendered the user continues to maintain focus and the animation continues to be user controlled.

5 Special Issues for Audio-based ODF Applications

It may seem counterintuitive, but audio is a primary medium for providing accessibility to many persons with disabilities. Whether it is an audio recording of someone reading text, or a real time computer generated "Text-To-Speech (TTS)," rendition, persons who are blind, or who live with severely impaired vision or a learning disabilities, often use audio as their primary reading modality. While we do not expect ODF applications should become audio recording and editing applications, there are nevertheless critical considerations that should be observed in order that documents produced by ODF applications might easily be used to create the smart audio renditions increasingly used by people who rely on audio renditions of textual content.

5.1 Where and How is Audio Used for Accessibility

Audio renditions of textual content are so common and powerful that they have been codified in an ANSI/NISO standard, Z39.86. This same specification has been adopted internationally by a consortium of libraries for the blind and print handicapped called the DAISY Consortium. In turn, this ANSI specification served as the basis for the U.S. legal mandate to provide accessible text books and curricular material in U.S. Schools, known as NIMAS.

Materials produced in audio include everything from novels for leisure reading, to newspapers, magazines, and technical reference material (in addition to curricular material). There are also national programs for creating and distributing such content across Europe, Canada, Australia, and Japan, as well as many other countries.

5.2 How ODF Fits In

ODF authoring applications are relevant to alternative media production, including audio, simply because alternative media such as audio renditions are just that,a different media edition of the same content produced for visual consumption by ODF applications:

5.3.1 Soft Page Breaks and Hard Page Numbering

[Ref to spec revision][Ref to spec revision]

Even the simplest content cannot be discussed in a group environment if there is no simple mechanism to point group members to a particular location in the document. Pagination is the most common resolution to this problem, however pagination is inextricably tied to a particular rendering of content. Alternative

media such as audio (and large print and braille) must have mechanisms to allow their users to know where they are relative to the hard page numbers of the primary source document. This scenario may be different if a group of visually impaired people have a sighted colleague with them!

5.3.2 Structural Markup

Effective use of audio renditions requires that users have the ability to move quickly back and forth through the audio rendition based on the structure of the document. Traditional audio playback equipment provided fast forward and rewind mechanisms, but these are highly inefficient because time offsets are actually irrelevant to content. What is relevant is the structure of the document? Does it have Chapters? Sub Sections? Footnotes? Sidebars? Paragraphs? Lists?

Effective support of alternative rendering, and especially audio rendering, requires that the source document be correctly tagged with structural markup. Indeed, the aforementioned ANSI and NIMAS specifications provide XML based markup to allow rendering agents to support quick movement forward and backward through content based on chapters, subsections, footnotes, paragraphs, and other structural elements. The most usable devices allow users to adjust "levels" of navigation, so that hierarchical structures such as X.Y.Z might be navigated at the X level, the Y level, or the Z level, at the user's option. This provides further emphasis on the importance of document structure when an ODF document is exported.

6 Glossary of Terms

Accessibility API

- an API (application programming interface) designed for assistive technologies to get information from an application and provide users audio and/or Braille outputs.
- Examples are Microsoft Active Accessibility(MSAA), Gnome Accessibility API, Java Accessibility API, etc.

Accessibility checker (evaluation tool)

- a software tool, which checks the accessibility of ODF files
- a user agent

Assistive Technology

- An application which assists users, who can not get access to standard user interfaces by providing input and output methods, such as audio, Braille, software keyboard, magnification. etc.

Converter

- a user agent which converts documents into ODF files or ODF files into other types of documents.
- a tool to help authors or users of other applications.

DAISY (Digital Accessible Information SYstem)

- a standard format for digital talking books.
- URL: "DAISY Consortium" http://www.daisy.org/

ODF Generator

- a user agent (software tool) on the client or server side that generates ODF files.
- an authoring tool.

Landmarks

- a concept for virtual landmarks used for non-visual navigation by using screen readers.
- ??TDB?? Could someone help me understand this please!

ODF editor.

- a user agent with editing capabilities for access and modification of ODF documents.
- an authoring tool.

ODF reader.

- a user agent without editing capabilities enabling people to access ODF content.

Relationships.

- The concept of a relationship between visual objects on a screen, necessary to understand the screen contents.
- ODF has functionality to add relationships between objects.
- E.g. form:for (See 11.5.7), draw:caption-id (See 9.2.15)

Synchronized media.

- a category of multimedia contents where
- various types of media content such as video, audio, text, and graphics are combined by using timing and synchronization controls.
- e.g. DAISY, SMIL (See http://www.w3.org/TR/SMIL/)

Screen magnifiers.

- a software program used to magnify any object on screen (e.g. characters, images, etc.)
- examples are office editor tools for low vision users.
- -an assistive technology.

Screen reader.

- a program used to read aloud from office editor screens or office reader screens for non-visual users.
- an assistive technology .

TTS (Text-to-Speech).

- A speech synthesis system: often called TTS because of its ability to convert text to speech.

User agent. (or ODF User Agent?)

- Any type of software tool which reads or writes ODF files

Voice office editor.

- a user agent with editing capabilities for users who prefer audio interactivity.
- an office editor.
- an assistive technology.
- an authoring tool.

Voice office reader

- a user agent without editing capabilities for users who prefer audio interactivity.
- an office reader.
- an assistive technology.

Appendix A. Acknowledgments

Contributors:

Stephen Noble, Design Science, Inc.
Chieko Asakawa, IBM
Pete Brunet, IBM
Hiro Takagi, IBM
Richard Schwerdtfeger, IBM
Janina Sajka, Individual
David Clark, Institute for Community Inclusion
David Pawson, Royal National Institute for the Blind
Peter Korn, Sun Microsystems, Inc.
Malte Timmermann, Sun Microsystems, Inc.

Appendix B. Notices

rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.