[contents]

Web Content Accessibility Guidelines (WCAG) 2.0

W3C Recommendation 11 December 2008

This version:
http://www.w3.org/TR/2008/REC-WCAG20-20081211/

Latest version:
http://www.w3.org/TR/WCAG20/

Previous version:
http://www.w3.org/TR/2008/PR-WCAG20-20081103/

Editors:
Ben Caldwell, Trace R&D Center, University of Wisconsin-Madison
Michael Cooper, W3C
Loretta Guarino Reid, Google, Inc.
Gregg Vanderheiden, Trace R&D Center, University of Wisconsin-Madison

Previous Editors:
Wendy Chisholm (until July 2006 while at W3C)
John Slatin (until June 2006 while at Accessibility Institute, University of Texas at Austin)
Jason White (until June 2005 while at University of Melbourne)

Please refer to the errata for this document, which may include normative corrections.

See also translations.

This document is also available in non-normative formats, available from Alternate Versions of Web Content Accessibility Guidelines 2.0.

Abstract

Web Content Accessibility Guidelines (WCAG) 2.0 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also

often make your Web content more usable to users in general.

WCAG 2.0 success criteria are written as testable statements that are not technology-specific. Guidance about satisfying the success criteria in specific technologies, as well as general information about interpreting the success criteria, is provided in separate documents. See Web Content Accessibility Guidelines (WCAG) Overview for an introduction and links to WCAG technical and educational material.

WCAG 2.0 succeeds Web Content Accessibility Guidelines 1.0 [WCAG10], which was published as a W3C Recommendation May 1999. Although it is possible to conform either to WCAG 1.0 or to WCAG 2.0 (or both), the W3C recommends that new and updated content use WCAG 2.0. The W3C also recommends that Web accessibility policies reference WCAG 2.0.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is the Web Content Accessibility Guidelines (WCAG) 2.0 W3C Recommendation from the Web Content Accessibility Guidelines Working Group.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

WCAG 2.0 is supported by the associated non-normative documents, Understanding WCAG 2.0 and Techniques for WCAG 2.0. Although those documents do not have the formal status that WCAG 2.0 itself has, they provide information important to understanding and implementing WCAG.

The Working Group requests that any comments be made using the provided online comment form. If this is not possible, comments can also be sent to public-comments-wcag20@w3.org. The archives for the public comments list are publicly available. Comments received on the WCAG 2.0 Recommendation cannot result in changes to this version of the guidelines, but may be addressed in errata or future versions of WCAG. The Working Group does not plan to make formal responses to comments. Archives of the WCAG WG mailing list discussions are publicly available, and future work undertaken by the Working Group may address comments received on this document.

This document has been produced as part of the W3C Web Accessibility Initiative (WAI). The goals of the WCAG Working Group are discussed in the WCAG Working Group charter. The WCAG Working Group is part of the WAI Technical Activity.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

Appendices

Introduction

This section is informative.

Web Content Accessibility Guidelines (WCAG) 2.0 defines how to make Web content more accessible to people with disabilities. Accessibility involves a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities. Although these guidelines cover a wide range of issues, they are not able to address the needs of people with all types, degrees, and combinations of disability. These guidelines also make Web content more usable by older individuals with changing abilities due to aging and often improve usability for users in general.

WCAG 2.0 is developed through the W3C process in cooperation with individuals and organizations around the world, with a goal of providing a shared standard for Web content accessibility that meets the needs of individuals, organizations, and governments internationally. WCAG 2.0 builds on WCAG 1.0 [WCAG10] and is designed to apply broadly to different Web technologies now and in the future, and to be testable with a combination of automated testing and human evaluation. For an introduction to WCAG, see the Web Content Accessibility Guidelines (WCAG) Overview.

Web accessibility depends not only on accessible content but also on accessible Web browsers and other user agents. Authoring tools also have an important role in Web accessibility. For an overview of how these components of Web development and interaction work together, see:

- Essential Components of Web Accessibility
- User Agent Accessibility Guidelines (UAAG) Overview
- Authoring Tool Accessibility Guidelines (ATAG) Overview

WCAG 2.0 Layers of Guidance

The individuals and organizations that use WCAG vary widely and include Web designers and developers, policy makers, purchasing agents, teachers, and students. In order to meet the varying needs of this audience, several layers of guidance are provided including overall *principles*, general *guidelines*, testable *success criteria* and a rich collection of *sufficient techniques, advisory techniques*, and *documented common failures* with examples, resource links and code.

- Principles - At the top are four principles that provide the foundation for Web accessibility: *perceivable, operable, understandable, and robust*. See also Understanding the Four Principles of Accessibility.
- Guidelines - Under the principles are guidelines. The 12 guidelines provide the basic goals that authors should work toward in order to make content more accessible to users with different disabilities. The guidelines are not testable, but provide the framework and overall objectives to help authors understand the success criteria and better implement the techniques.
- Success Criteria - For each guideline, testable success criteria are provided to allow WCAG 2.0 to be used where requirements and conformance testing are necessary such as in design specification, purchasing, regulation, and contractual agreements. In order to meet the needs of different groups and different situations, three levels of conformance are defined: A (lowest), AA, and AAA (highest). Additional information on WCAG levels can be found in Understanding Levels of Conformance.
- Sufficient and Advisory Techniques - For each of the *guidelines* and *success criteria* in the WCAG 2.0 document itself, the working group has also documented a wide variety of *techniques*. The techniques are informative and fall into two categories: those that are *sufficient* for meeting the success criteria and those that are *advisory*. The advisory techniques go beyond what is required by the individual success criteria and allow authors to better address the guidelines. Some advisory techniques address accessibility barriers that are not covered by the testable success criteria. Where common failures are known, these are also documented. See also Sufficient and Advisory Techniques in Understanding WCAG 2.0.

All of these layers of guidance (principles, guidelines, success criteria, and sufficient and advisory techniques) work together to provide guidance on how to make content more accessible. Authors are encouraged to view and apply all layers that they are able to, including the advisory techniques, in order to best address the needs of the widest possible range of users.

Note that even content that conforms at the highest level (AAA) will not be accessible to individuals with all types, degrees, or combinations of disability, particularly in the cognitive language and learning areas. Authors are encouraged to consider the full range of techniques, including the advisory techniques, as well as to seek relevant advice about current best practice to ensure that Web content is accessible, as far as possible, to this community. Metadata may assist users in finding content most suitable for their needs.

WCAG 2.0 Supporting Documents

The WCAG 2.0 document is designed to meet the needs of those who need a stable, referenceable technical standard. Other documents, called supporting documents, are based on the WCAG 2.0 document and address other important purposes, including the ability to be updated to describe how WCAG would be applied with new technologies. Supporting documents include:

1. How to Meet WCAG 2.0 - A customizable quick reference to WCAG 2.0 that includes all of the guidelines, success criteria, and techniques for authors to use as they are developing and evaluating Web content.
2. Understanding WCAG 2.0 - A guide to understanding and implementing WCAG 2.0. There is a short "Understanding" document for each guideline and success criterion in WCAG 2.0 as well as key topics.
3. Techniques for WCAG 2.0 - A collection of techniques and common failures, each in a separate document that includes a description, examples, code and tests.
4. The WCAG 2.0 Documents - A diagram and description of how the technical documents are related and linked.

See Web Content Accessibility Guidelines (WCAG) Overview for a description of the WCAG 2.0 supporting material, including education resources related to WCAG 2.0. Additional resources covering topics such as the business case for Web accessibility, planning implementation to improve the accessibility of Web sites, and accessibility policies are listed in WAI Resources.

Important Terms in WCAG 2.0

WCAG 2.0 includes three important terms that are different from WCAG 1.0. Each of these is introduced briefly below and defined more fully in the glossary.

Web Page

It is important to note that, in this standard, the term "Web page" includes much more than static HTML pages. It also includes the increasingly dynamic Web pages that are emerging on the Web, including "pages" that can present entire virtual interactive communities. For example, the term "Web page" includes an immersive, interactive movie-like experience found at a single URI. For more information, see Understanding "Web Page".

Programmatically Determined

Several success criteria require that content (or certain aspects of content) can be "programmatically determined." This means that the content is delivered in such a way that user agents, including assistive technologies, can extract and present this information to users in different modalities. For more information, see Understanding Programmatically Determined.

Accessibility Supported

Using a technology in a way that is accessibility supported means that it works with assistive technologies (AT) and the accessibility features of operating systems, browsers, and other user agents. Technology features can only be relied upon to conform to WCAG 2.0 success criteria if they are used in a way that is "accessibility supported". Technology features can be used in ways that are not accessibility

supported (do not work with assistive technologies, etc.) as long as they are not relied upon to conform to any success criterion (i.e., the same information or functionality is also available another way that is supported).
The definition of "accessibility supported" is provided in the Appendix A: Glossary section of these guidelines. For more information, see Understanding Accessibility Support.

WCAG 2.0 Guidelines

This section is normative.

Principle 1: Perceivable - Information and user interface components must be presentable to users in ways they can perceive.

Guideline 1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

Understanding Guideline 1.1

1.1.1 Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A)

How to Meet 1.1.1
Understanding 1.1.1

- Controls, Input: If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to Guideline 4.1 for additional requirements for controls and content that accepts user input.)
- Time-Based Media: If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to Guideline 1.2 for additional requirements for media.)
- Test: If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content.
- Sensory: If non-text content is primarily intended to create a specific sensory experience, then text alternatives at least provide descriptive identification of the non-text content.
- CAPTCHA: If the purpose of non-text content is to confirm

that content is being accessed by a person rather than a computer, then text alternatives that identify and describe the purpose of the non-text content are provided, and alternative forms of CAPTCHA using output modes for different types of sensory perception are provided to accommodate different disabilities.

- Decoration, Formatting, Invisible: If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.

Guideline 1.2 Time-based Media: Provide alternatives for time-based media.

Understanding Guideline 1.2

1.2.1 Audio-only and Video-only (Prerecorded): For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such: (Level A)

- Prerecorded Audio-only: An alternative for time-based media is provided that presents equivalent information for prerecorded audio-only content.
- Prerecorded Video-only: Either an alternative for time-based media or an audio track is provided that presents equivalent information for prerecorded video-only content.

How to Meet 1.2.1
Understanding 1.2.1

1.2.2 Captions (Prerecorded): Captions are provided for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A)

How to Meet 1.2.2
Understanding 1.2.2

1.2.3 Audio Description or Media Alternative (Prerecorded): An alternative for time-based media or audio description of the prerecorded video content is provided for synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A)

How to Meet 1.2.3
Understanding 1.2.3

1.2.4 Captions (Live): Captions are provided for all live audio content in synchronized media. (Level AA)

How to Meet 1.2.4
Understanding 1.2.4

1.2.5 Audio Description (Prerecorded): Audio description is provided for all prerecorded video content in synchronized media. (Level AA)

How to Meet 1.2.5
Understanding 1.2.5

1.2.6 Sign Language (Prerecorded): Sign language interpretation is provided for all prerecorded audio content in synchronized media. (Level AAA)

How to Meet 1.2.6
Understanding 1.2.6

1.2.7 Extended Audio Description (Prerecorded): Where pauses in foreground audio are insufficient to allow audio descriptions to convey the sense of the video, extended audio description is provided for all prerecorded video content in synchronized media. (Level AAA)

How to Meet 1.2.7
Understanding 1.2.7

1.2.8 Media Alternative (Prerecorded): An alternative for time-based media is provided for all prerecorded synchronized media and for all prerecorded video-only media. (Level AAA)

How to Meet 1.2.8
Understanding 1.2.8

1.2.9 Audio-only (Live): An alternative for time-based media that presents equivalent information for live audio-only content is provided. (Level AAA)

How to Meet 1.2.9
Understanding 1.2.9

Guideline 1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

Understanding Guideline 1.3

1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)

How to Meet 1.3.1
Understanding 1.3.1

1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)

How to Meet 1.3.2
Understanding 1.3.2

1.3.3 Sensory Characteristics: Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, size,

How to Meet 1.3.3
Understanding 1.3.3

visual location, orientation, or sound. (Level A)

Note: For requirements related to color, refer to Guideline 1.4.

Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Understanding Guideline 1.4

1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)

How to Meet 1.4.1
Understanding 1.4.1

Note: This success criterion addresses color perception specifically. Other forms of perception are covered in Guideline 1.3 including programmatic access to color and other visual presentation coding.

1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A)

How to Meet 1.4.2
Understanding 1.4.2

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether or not it is used to meet other success criteria) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA)

How to Meet 1.4.3
Understanding 1.4.3

- Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;
- Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
- Logotypes: Text that is part of a logo or brand name has no minimum contrast requirement.

1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA)

How to Meet 1.4.4
Understanding 1.4.4

1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA)

- Customizable: The image of text can be visually customized to the user's requirements;
- Essential: A particular presentation of text is essential to the information being conveyed.

Note: Logotypes (text that is part of a logo or brand name) are considered essential.

How to Meet 1.4.5
Understanding 1.4.5

1.4.6 Contrast (Enhanced): The visual presentation of text and images of text has a contrast ratio of at least 7:1, except for the following: (Level AAA)

- Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 4.5:1;
- Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
- Logotypes: Text that is part of a logo or brand name has no minimum contrast requirement.

How to Meet 1.4.6
Understanding 1.4.6

1.4.7 Low or No Background Audio: For prerecorded audio-only content that (1) contains primarily speech in the foreground, (2) is not an audio CAPTCHA or audio logo, and (3) is not vocalization intended to be primarily musical expression such as singing or rapping, at least one of the following is true: (Level AAA)

- No Background: The audio does not contain background sounds.
- Turn Off: The background sounds can be turned off.
- 20 dB: The background sounds are at least 20 decibels lower than the foreground speech content, with the exception of occasional sounds that last for only one or two seconds.

How to Meet 1.4.7
Understanding 1.4.7

Note: Per the definition of "decibel," background sound that meets this requirement will be approximately four times quieter than the foreground speech content.

1.4.8 Visual Presentation: For the visual presentation of blocks of text, a mechanism is available to achieve the following: (Level AAA)

How to Meet 1.4.8
Understanding 1.4.8

1. Foreground and background colors can be selected by the user.
2. Width is no more than 80 characters or glyphs (40 if CJK).
3. Text is not justified (aligned to both the left and the right margins).
4. Line spacing (leading) is at least space-and-a-half within paragraphs, and paragraph spacing is at least 1.5 times larger than the line spacing.
5. Text can be resized without assistive technology up to 200 percent in a way that does not require the user to scroll horizontally to read a line of text on a full-screen window.

1.4.9 Images of Text (No Exception): Images of text are only used for pure decoration or where a particular presentation of text is essential to the information being conveyed. (Level AAA)

How to Meet 1.4.9
Understanding 1.4.9

Note: Logotypes (text that is part of a logo or brand name) are considered essential.

Principle 2: Operable - User interface components and navigation must be operable.

Guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard.

Understanding Guideline 2.1

2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)

How to Meet 2.1.1
Understanding 2.1.1

Note 1: This exception relates to the underlying function, not the input technique. For example, if using handwriting to enter text, the input technique (handwriting) requires path-dependent input but the underlying function (text input) does not.

Note 2: This does not forbid and should not discourage providing mouse input or other input methods in addition to keyboard operation.

2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)

How to Meet 2.1.2
Understanding 2.1.2

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)

How to Meet 2.1.3
Understanding 2.1.3

Guideline 2.2 Enough Time: Provide users enough time to read and use content.

Understanding Guideline 2.2

2.2.1 Timing Adjustable: For each time limit that is set by the content, at least one of the following is true: (Level A)

- Turn off: The user is allowed to turn off the time limit before encountering it; or
- Adjust: The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or
- Extend: The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or
- Real-time Exception: The time limit is a required part of

How to Meet 2.2.1
Understanding 2.2.1

a real-time event (for example, an auction), and no alternative to the time limit is possible; or
- Essential Exception: The time limit is essential and extending it would invalidate the activity; or
- 20 Hour Exception: The time limit is longer than 20 hours.

Note: This success criterion helps ensure that users can complete tasks without unexpected changes in content or context that are a result of a time limit. This success criterion should be considered in conjunction with Success Criterion 3.2.1, which puts limits on changes of content or context as a result of user action.

2.2.2 Pause, Stop, Hide: For moving, blinking, scrolling, or auto-updating information, all of the following are true: (Level A)

How to Meet 2.2.2
Understanding 2.2.2

- Moving, blinking, scrolling: For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is essential; and
- Auto-updating: For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

Note 1: For requirements related to flickering or flashing content, refer to Guideline 2.3.

Note 2: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

Note 3: Content that is updated periodically by software or that is streamed to the user agent is not required to preserve or present information that is generated or received between the initiation of the pause and resuming presentation, as this may not be technically possible, and in many situations could be misleading to do so.

Note 4: An animation that occurs as part of a preload phase or similar situation can be considered essential if interaction cannot occur during that phase for all users and if not indicating progress could confuse users or cause them to think that content was frozen or broken.

2.2.3 No Timing: Timing is not an essential part of the event or activity presented by the content, except for non-interactive synchronized media and real-time events. (Level AAA)

How to Meet 2.2.3
Understanding 2.2.3

2.2.4 Interruptions: Interruptions can be postponed or suppressed by the user, except interruptions involving an emergency. (Level AAA)

How to Meet 2.2.4
Understanding 2.2.4

2.2.5 Re-authenticating: When an authenticated session expires, the user can continue the activity without loss of data after re-authenticating. (Level AAA)

How to Meet 2.2.5
Understanding 2.2.5

Guideline 2.3 Seizures: Do not design content in a way that is known to cause seizures.

Understanding Guideline 2.3

2.3.1 Three Flashes or Below Threshold: Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds. (Level A)

How to Meet 2.3.1
Understanding 2.3.1

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

2.3.2 Three Flashes: Web pages do not contain anything that flashes more than three times in any one second period. (Level AAA)

How to Meet 2.3.2
Understanding 2.3.2

Guideline 2.4 Navigable: Provide ways to

Understanding Guideline 2.4

help users navigate, find content, and determine where they are.

2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A)

How to Meet 2.4.1
Understanding 2.4.1

2.4.2 Page Titled: Web pages have titles that describe topic or purpose. (Level A)

How to Meet 2.4.2
Understanding 2.4.2

2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)

How to Meet 2.4.3
Understanding 2.4.3

2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A)

How to Meet 2.4.4
Understanding 2.4.4

2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA)

How to Meet 2.4.5
Understanding 2.4.5

2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)

How to Meet 2.4.6
Understanding 2.4.6

2.4.7 Focus Visible: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA)

How to Meet 2.4.7
Understanding 2.4.7

2.4.8 Location: Information about the user's location within a set of Web pages is available. (Level AAA)

How to Meet 2.4.8
Understanding 2.4.8

2.4.9 Link Purpose (Link Only): A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general. (Level AAA)

How to Meet 2.4.9
Understanding 2.4.9

2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA)

How to Meet 2.4.10
Understanding 2.4.10

Note 1: "Heading" is used in its general sense and includes titles and other ways to add a heading to different types of content.

Note 2: This success criterion covers sections within writing, not user interface components. User Interface components are covered under Success Criterion 4.1.2.

Principle 3: Understandable - Information and the operation of user interface must be understandable.

Guideline 3.1 Readable: Make text content readable and understandable.

Understanding Guideline 3.1

3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A)

How to Meet 3.1.1
Understanding 3.1.1

3.1.2 Language of Parts: The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text. (Level AA)

How to Meet 3.1.2
Understanding 3.1.2

3.1.3 Unusual Words: A mechanism is available for identifying specific definitions of words or phrases used in an unusual or restricted way, including idioms and jargon. (Level AAA)

How to Meet 3.1.3
Understanding 3.1.3

3.1.4 Abbreviations: A mechanism for identifying the expanded form or meaning of abbreviations is available. (Level AAA)

How to Meet 3.1.4
Understanding 3.1.4

3.1.5 Reading Level: When text requires reading ability more

How to Meet 3.1.5

advanced than the lower secondary education level after removal of proper names and titles, supplemental content, or a version that does not require reading ability more advanced than the lower secondary education level, is available. (Level AAA)

Understanding 3.1.5

3.1.6 Pronunciation: A mechanism is available for identifying specific pronunciation of words where meaning of the words, in context, is ambiguous without knowing the pronunciation. (Level AAA)

How to Meet 3.1.6
Understanding 3.1.6

Guideline 3.2 Predictable: Make Web pages appear and operate in predictable ways.

Understanding Guideline 3.2

3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A)

How to Meet 3.2.1
Understanding 3.2.1

3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)

How to Meet 3.2.2
Understanding 3.2.2

3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)

How to Meet 3.2.3
Understanding 3.2.3

3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)

How to Meet 3.2.4
Understanding 3.2.4

3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA)

How to Meet 3.2.5
Understanding 3.2.5

Guideline 3.3 Input Assistance: Help users

Understanding Guideline 3.3

avoid and correct mistakes.

3.3.1 Error Identification: If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text. (Level A)

How to Meet 3.3.1
Understanding 3.3.1

3.3.2 Labels or Instructions: Labels or instructions are provided when content requires user input. (Level A)

How to Meet 3.3.2
Understanding 3.3.2

3.3.3 Error Suggestion: If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content. (Level AA)

How to Meet 3.3.3
Understanding 3.3.3

3.3.4 Error Prevention (Legal, Financial, Data): For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true: (Level AA)

1. Reversible: Submissions are reversible.
2. Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.
3. Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

How to Meet 3.3.4
Understanding 3.3.4

3.3.5 Help: Context-sensitive help is available. (Level AAA)

How to Meet 3.3.5
Understanding 3.3.5

3.3.6 Error Prevention (All): For Web pages that require the user to submit information, at least one of the following is true: (Level AAA)

1. Reversible: Submissions are reversible.
2. Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.
3. Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

How to Meet 3.3.6
Understanding 3.3.6

Principle 4: Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

Guideline 4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies.

Understanding Guideline 4.1

4.1.1 Parsing: In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A)

How to Meet 4.1.1
Understanding 4.1.1

Note: Start and end tags that are missing a critical character in their formation, such as a closing angle bracket or a mismatched attribute value quotation mark are not complete.

4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)

How to Meet 4.1.2
Understanding 4.1.2

Note: This success criterion is primarily for Web authors who develop or script their own user interface components. For example, standard HTML controls already meet this success criterion when used according to specification.

Conformance

This section is normative.

This section lists requirements for conformance to WCAG 2.0. It also gives information about how to make conformance claims, which are optional. Finally, it describes what it means to be accessibility supported, since only accessibility-supported ways of using technologies can be relied upon for conformance. Understanding Conformance includes further explanation of the accessibility-supported concept.

Conformance Requirements

In order for a Web page to conform to WCAG 2.0, all of the following conformance requirements must be satisfied:

1. Conformance Level: One of the following levels of conformance is met in full.
   - Level A: For Level A conformance (the minimum level of conformance), the Web page satisfies all the Level A Success Criteria, or a conforming alternate version is provided.
   - Level AA: For Level AA conformance, the Web page satisfies all the Level A and Level AA Success Criteria, or a Level AA conforming alternate version is provided.
   - Level AAA: For Level AAA conformance, the Web page satisfies all the Level A, Level AA and Level AAA Success Criteria, or a Level AAA conforming alternate version is provided.

   Note 1: Although conformance can only be achieved at the stated levels, authors are encouraged to report (in their claim) any progress toward meeting success criteria from all levels beyond the achieved level of conformance.

   Note 2: It is not recommended that Level AAA conformance be required as a general policy for entire sites because it is not possible to satisfy all Level AAA Success Criteria for some content.

2. Full pages: Conformance (and conformance level) is for full Web page(s) only, and cannot be achieved if part of a Web page is excluded.

   Note 1: For the purpose of determining conformance, alternatives to part of a page's content are considered part of the page when the alternatives can be obtained directly from the page, e.g., a long description or an alternative presentation of a video.

   Note 2: Authors of Web pages that cannot conform due to content outside of the author's control may consider a Statement of Partial Conformance.

3. Complete processes: When a Web page is one of a series of Web pages presenting a process (i.e., a sequence of steps that need to be completed in order to accomplish an activity), all Web pages in the process conform at the specified level or better. (Conformance is not possible at a particular level if any page in the process does not conform at that level or better.)

   Example: An online store has a series of pages that are used to select and purchase products. All pages in the series from start to finish (checkout) conform in order for any page that is part of the process to conform.

4. Only Accessibility-Supported Ways of Using Technologies: Only accessibility-supported ways of using technologies are relied upon to satisfy the success criteria. Any information or functionality that is provided in a way that is not accessibility supported is also available in a way that is accessibility supported. (See Understanding accessibility support.)

5. Non-Interference: If technologies are used in a way that is not accessibility supported,

or if they are used in a non-conforming way, then they do not block the ability of users to access the rest of the page. In addition, the Web page as a whole continues to meet the conformance requirements under each of the following conditions:

1. when any technology that is not relied upon is turned on in a user agent,
2. when any technology that is not relied upon is turned off in a user agent, and
3. when any technology that is not relied upon is not supported by a user agent

In addition, the following success criteria apply to all content on the page, including content that is not otherwise relied upon to meet conformance, because failure to meet them could interfere with any use of the page:

- 1.4.2 - Audio Control,
- 2.1.2 - No Keyboard Trap,
- 2.3.1 - Three Flashes or Below Threshold, and
- 2.2.2 - Pause, Stop, Hide.

Note: If a page cannot conform (for example, a conformance test page or an example page), it cannot be included in the scope of conformance or in a conformance claim.

For more information, including examples, see Understanding Conformance Requirements.

Conformance Claims (Optional)

Conformance is defined only for Web pages. However, a conformance claim may be made to cover one page, a series of pages, or multiple related Web pages.

Required Components of a Conformance Claim

Conformance claims are not required. Authors can conform to WCAG 2.0 without making a claim. However, if a conformance claim is made, then the conformance claim must include the following information:

1. Date of the claim
2. Guidelines title, version and URI "Web Content Accessibility Guidelines 2.0 at http://www.w3.org/TR/2008/REC-WCAG20-20081211/"
3. Conformance level satisfied: (Level A, AA or AAA)
4. A concise description of the Web pages, such as a list of URIs for which the claim is made, including whether subdomains are included in the claim.

   Note 1: The Web pages may be described by list or by an expression that describes all of the URIs included in the claim.

   Note 2: Web-based products that do not have a URI prior to installation on the customer's Web site may have a statement that the product would conform when installed.
5. A list of the Web content technologies relied upon.

Note: If a conformance logo is used, it would constitute a claim and must be accompanied by the required components of a conformance claim listed above.

Optional Components of a Conformance Claim

In addition to the required components of a conformance claim above, consider providing additional information to assist users. Recommended additional information includes:

- A list of success criteria beyond the level of conformance claimed that have been met. This information should be provided in a form that users can use, preferably machine-readable metadata.
- A list of the specific technologies that are "*used but not relied upon*."
- A list of user agents, including assistive technologies that were used to test the content.
- Information about any additional steps taken that go beyond the success criteria to enhance accessibility.
- A machine-readable metadata version of the list of specific technologies that are relied upon.
- A machine-readable metadata version of the conformance claim.

Note 1: Refer to Understanding Conformance Claims for more information and example conformance claims.

Note 2: Refer to Understanding Metadata for more information about the use of metadata in conformance claims.

Statement of Partial Conformance - Third Party Content

Sometimes, Web pages are created that will later have additional content added to them. For example, an email program, a blog, an article that allows users to add comments, or applications supporting user-contributed content. Another example would be a page, such as a portal or news site, composed of content aggregated from multiple contributors, or sites that automatically insert content from other sources over time, such as when advertisements are inserted dynamically.

In these cases, it is not possible to know at the time of original posting what the uncontrolled content of the pages will be. It is important to note that the uncontrolled content can affect the accessibility of the controlled content as well. Two options are available:

1. A determination of conformance can be made based on best knowledge. If a page of this type is monitored and repaired (non-conforming content is removed or brought into conformance) within two business days, then a determination or claim of conformance can be made since, except for errors in externally contributed content which are corrected or removed when encountered, the page conforms. No conformance claim can be made if it is not possible to monitor or correct non-conforming content;
   OR
2. A "statement of partial conformance" may be made that the page does not conform, but could conform if certain parts were removed. The form of that statement would be, "This

page does not conform, but would conform to WCAG 2.0 at level X if the following parts from uncontrolled sources were removed." In addition, the following would also be true of uncontrolled content that is described in the statement of partial conformance:

a. It is not content that is under the author's control.
b. It is described in a way that users can identify (e.g., they cannot be described as "all parts that we do not control" unless they are clearly marked as such.)

Statement of Partial Conformance - Language

A "statement of partial conformance due to language" may be made when the page does not conform, but would conform if accessibility support existed for (all of) the language(s) used on the page. The form of that statement would be, "This page does not conform, but would conform to WCAG 2.0 at level X if accessibility support existed for the following language(s):"

Appendix A: Glossary

This section is normative.

abbreviation

shortened form of a word, phrase, or name where the abbreviation has not become part of the language

Note 1: This includes initialisms and acronyms where:

1. initialisms are shortened forms of a name or phrase made from the initial letters of words or syllables contained in that name or phrase

   Note 1: Not defined in all languages.

   Example 1: SNCF is a French initialism that contains the initial letters of the Société Nationale des Chemins de Fer, the French national railroad.

   Example 2: ESP is an initialism for extrasensory perception.

2. acronyms are abbreviated forms made from the initial letters or parts of other words (in a name or phrase) which may be pronounced as a word

   Example: NOAA is an acronym made from the initial letters of the National Oceanic and Atmospheric Administration in the United States.

Note 2: Some companies have adopted what used to be an initialism as their company name. In these cases, the new name of the company is the letters (for example, Ecma) and the word is no longer considered an abbreviation.

accessibility supported

supported by users' assistive technologies as well as the accessibility features in browsers and other user agents

To qualify as an accessibility-supported use of a Web content technology (or feature of a technology), both 1 and 2 must be satisfied for a Web content technology (or feature):

1. The way that the Web content technology is used must be

supported by users' assistive technology (AT). This means that the way that the technology is used has been tested for interoperability with users' assistive technology in the human language(s) of the content,
AND

2. The Web content technology must have accessibility-supported user agents that are available to users. This means that at least one of the following four statements is true:
    a. The technology is supported natively in widely-distributed user agents that are also accessibility supported (such as HTML and CSS);
    OR
    b. The technology is supported in a widely-distributed plug-in that is also accessibility supported;
    OR
    c. The content is available in a closed environment, such as a university or corporate network, where the user agent required by the technology and used by the organization is also accessibility supported;
    OR
    d. The user agent(s) that support the technology are accessibility supported and are available for download or purchase in a way that:
        - does not cost a person with a disability any more than a person without a disability and
        - is as easy to find and obtain for a person with a disability as it is for a person without disabilities.

Note 1: The WCAG Working group and the W3C do not specify which or how much support by assistive technologies there must be for a particular use of a Web technology in order for it to be classified as accessibility supported. (See Level of Assistive Technology Support Needed for "Accessibility Support".)

Note 2: Web technologies can be used in ways that are not accessibility supported as long as they are not relied upon and the page as a whole meets the conformance requirements, including Conformance Requirement 4: Only Accessibility-Supported Ways of Using Technologies and Conformance Requirement 5: Non-Interference, are met.

Note 3: When a Web Technology is used in a way that is "accessibility supported," it does not imply that the entire technology or all uses of the technology are supported. Most technologies, including HTML, lack support for at least one feature or use. Pages conform to WCAG only if the uses of the technology that are accessibility supported can be relied upon to meet WCAG requirements.

Note 4: When citing Web content technologies that have multiple versions, the version(s) supported should be specified.

Note 5: One way for authors to locate uses of a technology that are accessibility supported would be to consult compilations of uses that are documented to be accessibility supported. (See Understanding Accessibility-Supported Web Technology

Uses.) Authors, companies, technology vendors, or others may document accessibility-supported ways of using Web content technologies. However, all ways of using technologies in the documentation would need to meet the definition of accessibility-supported Web content technologies above.

alternative for time-based media

document including correctly sequenced text descriptions of time-based visual and auditory information and providing a means for achieving the outcomes of any time-based interaction

Note: A screenplay used to create the synchronized media content would meet this definition only if it was corrected to accurately represent the final synchronized media after editing.

ambiguous to users in general

the purpose cannot be determined from the link and all information of the Web page presented to the user simultaneously with the link (i.e., readers without disabilities would not know what a link would do until they activated it)

Example: The word guava in the following sentence "One of the notable exports is guava" is a link. The link could lead to a definition of guava, a chart listing the quantity of guava exported or a photograph of people harvesting guava. Until the link is activated, all readers are unsure and the person with a disability is not at any disadvantage.

ASCII art

picture created by a spatial arrangement of characters or glyphs (typically from the 95 printable characters defined by ASCII).

assistive technology (as used in this document)

hardware and/or software that acts as a user agent, or along with a mainstream user agent, to provide functionality to meet the requirements of users with disabilities that go beyond those offered by mainstream user agents

Note 1: functionality provided by assistive technology includes alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).

Note 2: Assistive technologies often communicate data and messages with mainstream user agents by using and monitoring APIs.

Note 3: The distinction between mainstream user agents and assistive technologies is not absolute. Many mainstream user agents provide some features to assist individuals with disabilities. The basic difference is that mainstream user agents target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The mainstream user agent may provide important functionality to assistive technologies like retrieving Web content from program objects or parsing markup into identifiable bundles.

Example: Assistive technologies that are important in the context of this document include the following:

- screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;
- screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;
- text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;
- speech recognition software, which may be used by people who have some physical disabilities;
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use head pointers, single switches, sip/puff and other special input devices.);
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

audio
: the technology of sound reproduction

Note: Audio can be created synthetically (including speech synthesis), recorded from real world sounds, or both.

audio description
: narration added to the soundtrack to describe important visual details that cannot be understood from the main soundtrack alone

Note 1: Audio description of video provides information about actions, characters, scene changes, on-screen text, and other visual content.

Note 2: In standard audio description, narration is added during existing pauses in dialogue. (See also extended audio description.)

Note 3: Where all of the video information is already provided in existing audio, no additional audio description is necessary.

Note 4: Also called "video description" and "descriptive narration."

audio-only
: a time-based presentation that contains only audio (no video and no interaction)

blinking
: switch back and forth between two visual states in a way that is meant to draw attention

Note: See also flash. It is possible for something to be large enough and blink brightly enough at the right frequency to be also classified as a flash.

blocks of text
: more than one sentence of text

CAPTCHA
: initialism for "Completely Automated Public Turing test to tell Computers and Humans Apart"

  Note 1: CAPTCHA tests often involve asking the user to type in text that is displayed in an obscured image or audio file.

  Note 2: A Turing test is any system of tests designed to differentiate a human from a computer. It is named after famed computer scientist Alan Turing. The term was coined by researchers at Carnegie Mellon University. [CAPTCHA]

captions
: synchronized visual and/or text alternative for both speech and non-speech audio information needed to understand the media content

  Note 1: Captions are similar to dialogue-only subtitles except captions convey not only the content of spoken dialogue, but also equivalents for non-dialogue audio information needed to understand the program content, including sound effects, music, laughter, speaker identification and location.

  Note 2: Closed Captions are equivalents that can be turned on and off with some players.

  Note 3: Open Captions are any captions that cannot be turned off. For example, if the captions are visual equivalent images of text embedded in video.

  Note 4: Captions should not obscure or obstruct relevant information in the video.

  Note 5: In some countries, captions are called subtitles.

  Note 6: Audio descriptions can be, but do not need to be, captioned since they are descriptions of information that is already presented visually.

changes of context
: major changes in the content of the Web page that, if made without user awareness, can disorient users who are not able to view the entire page simultaneously
  Changes in context include changes of:

  1. user agent;
  2. viewport;
  3. focus;
  4. content that changes the meaning of the Web page.

  Note: A change of content is not always a change of context. Changes in content, such as an expanding outline, dynamic menu, or a tab control do not necessarily change the context, unless they also change one of the above (e.g., focus).

  Example: Opening a new window, moving focus to a different component, going to a new page (including anything that would look to a user as if they had moved to a new page) or significantly re-arranging the content of a page are examples of changes of context.

conformance
: satisfying all the requirements of a given standard, guideline or specification

conforming alternate version
: version that

1. conforms at the designated level, and
2. provides all of the same information and functionality in the same human language, and
3. is as up to date as the non-conforming content, and
4. for which at least one of the following is true:
    a. the conforming version can be reached from the non-conforming page via an accessibility-supported mechanism, or
    b. the non-conforming version can only be reached from the conforming version, or
    c. the non-conforming version can only be reached from a conforming page that also provides a mechanism to reach the conforming version

Note 1: In this definition, "can only be reached" means that there is some mechanism, such as a conditional redirect, that prevents a user from "reaching" (loading) the non-conforming page unless the user had just come from the conforming version.

Note 2: The alternate version does not need to be matched page for page with the original (e.g., the conforming alternate version may consist of multiple pages).

Note 3: If multiple language versions are available, then conforming alternate versions are required for each language offered.

Note 4: Alternate versions may be provided to accommodate different technology environments or user groups. Each version should be as conformant as possible. One version would need to be fully conformant in order to meet conformance requirement 1.

Note 5: The conforming alternative version does not need to reside within the scope of conformance, or even on the same Web site, as long as it is as freely available as the non-conforming version.

Note 6: Alternate versions should not be confused with supplementary content, which support the original page and enhance comprehension.

Note 7: Setting user preferences within the content to produce a conforming version is an acceptable mechanism for reaching another version as long as the method used to set the preferences is accessibility supported.

See Understanding Conforming Alternate Versions

content (Web content)
: information and sensory experience to be communicated to the user by means of a user agent, including code or markup that defines the content's structure, presentation, and interactions

context-sensitive help

help text that provides information related to the function currently being performed

Note: Clear labels can act as context-sensitive help.

contrast ratio

(L1 + 0.05) / (L2 + 0.05), where

- L1 is the relative luminance of the lighter of the colors, and
- L2 is the relative luminance of the darker of the colors.

Note 1: Contrast ratios can range from 1 to 21 (commonly written 1:1 to 21:1).

Note 2: Because authors do not have control over user settings as to how text is rendered (for example font smoothing or anti-aliasing), the contrast ratio for text can be evaluated with anti-aliasing turned off.

Note 3: For the purpose of Success Criteria 1.4.3 and 1.4.6, contrast is measured with respect to the specified background over which the text is rendered in normal usage. If no background color is specified, then white is assumed.

Note 4: Background color is the specified color of content over which the text is to be rendered in normal usage. It is a failure if no background color is specified when the text color is specified, because the user's default background color is unknown and cannot be evaluated for sufficient contrast. For the same reason, it is a failure if no text color is specified when a background color is specified.

Note 5: When there is a border around the letter, the border can add contrast and would be used in calculating the contrast between the letter and its background. A narrow border around the letter would be used as the letter. A wide border around the letter that fills in the inner details of the letters acts as a halo and would be considered background.

Note 6: WCAG conformance should be evaluated for color pairs specified in the content that an author would expect to appear adjacent in typical presentation. Authors need not consider unusual presentations, such as color changes made by the user agent, except where caused by authors' code.

correct reading sequence

any sequence where words and paragraphs are presented in an order that does not change the meaning of the content

emergency

a sudden, unexpected situation or occurrence that requires immediate action to preserve health, safety, or property

essential

if removed, would fundamentally change the information or functionality of the content, and information and functionality cannot be achieved in another way that would conform

extended audio description

audio description that is added to an audiovisual presentation by pausing the video so that there is time to add additional description

Note: This technique is only used when the sense of the video would be lost without the additional audio description and the pauses between dialogue/narration are too short.

flash
: a pair of opposing changes in relative luminance that can cause seizures in some people if it is large enough and in the right frequency range

Note 1: See general flash and red flash thresholds for information about types of flash that are not allowed.

Note 2: See also blinking.

functionality
: processes and outcomes achievable through user action

general flash and red flash thresholds
: a flash or rapidly changing image sequence is below the threshold (i.e., content passes) if any of the following are true:

1. there are no more than three general flashes and / or no more than three red flashes within any one-second period; or
2. the combined area of flashes occurring concurrently occupies no more than a total of .006 steradians within any 10 degree visual field on the screen (25% of any 10 degree visual field on the screen) at typical viewing distance

where:

- A general flash is defined as a pair of opposing changes in relative luminance of 10% or more of the maximum relative luminance where the relative luminance of the darker image is below 0.80; and where "a pair of opposing changes" is an increase followed by a decrease, or a decrease followed by an increase, and
- A red flash is defined as any pair of opposing transitions involving a saturated red.

Exception: Flashing that is a fine, balanced, pattern such as white noise or an alternating checkerboard pattern with "squares" smaller than 0.1 degree (of visual field at typical viewing distance) on a side does not violate the thresholds.

Note 1: For general software or Web content, using a 341 x 256 pixel rectangle anywhere on the displayed screen area when the content is viewed at 1024 x 768 pixels will provide a good estimate of a 10 degree visual field for standard screen sizes and viewing distances (e.g., 15-17 inch screen at 22-26 inches). (Higher resolutions displays showing the same rendering of the content yield smaller and safer images so it is lower resolutions that are used to define the thresholds.)

Note 2: A transition is the change in relative luminance (or relative luminance/color for red flashing) between adjacent peaks and valleys in a plot of relative luminance (or relative luminance/color for red flashing) measurement against time. A flash consists of two opposing transitions.

Note 3: The current working definition in the field for "pair of opposing transitions involving a saturated red" is where, for either or both states involved in each transition, R/(R+ G + B) >= 0.8, and the change in the value of (R-G-B)x320 is > 20 (negative values of (R-G-B)x320 are set to zero) for both transitions. R, G, B values range from 0-1 as specified in “relative luminance” definition. [HARDING-BINNIE]

Note 4: Tools are available that will carry out analysis from video screen capture. However, no tool is necessary to evaluate for this condition if flashing is less than or equal to 3 flashes in any one second. Content automatically passes (see #1 and #2 above).

human language

language that is spoken, written or signed (through visual or tactile means) to communicate with humans

Note: See also sign language.

idiom

phrase whose meaning cannot be deduced from the meaning of the individual words and the specific words cannot be changed without losing the meaning

Note: idioms cannot be translated directly, word for word, without losing their (cultural or language-dependent) meaning.

Example 1: In English, "spilling the beans" means "revealing a secret." However, "knocking over the beans" or "spilling the vegetables" does not mean the same thing.

Example 2: In Japanese, the phrase "さじを投げる" literally translates into "he throws a spoon," but it means that there is nothing he can do and finally he gives up.

Example 3: In Dutch, "Hij ging met de kippen op stok" literally translates into "He went to roost with the chickens," but it means that he went to bed early.

image of text

text that has been rendered in a non-text form (e.g., an image) in order to achieve a particular visual effect

Note: This does not include text that is part of a picture that contains significant other visual content.

Example: A person's name on a nametag in a photograph.

informative

for information purposes and not required for conformance

Note: Content required for conformance is referred to as "normative."

input error

information provided by the user that is not accepted

Note: This includes:

1. Information that is required by the Web page but omitted by the user
2. Information that is provided by the user but that falls outside the required data format or values

jargon

words used in a particular way by people in a particular field

Example: The word StickyKeys is jargon from the field of assistive technology/accessibility.

keyboard interface

interface used by software to obtain keystroke input

Note 1: A keyboard interface allows users to provide keystroke input to programs even if the native technology does not contain a keyboard.

Example: A touchscreen PDA has a keyboard interface built into its operating system as well as a connector for external keyboards. Applications on the PDA can use the interface to obtain keyboard input either from an external keyboard or from other applications that provide simulated keyboard output, such as handwriting interpreters or speech-to-text applications with "keyboard emulation" functionality.

Note 2: Operation of the application (or parts of the application) through a keyboard-operated mouse emulator, such as MouseKeys, does not qualify as operation through a keyboard interface because operation of the program is through its pointing device interface, not through its keyboard interface.

label

text or other component with a text alternative that is presented to a user to identify a component within Web content

Note 1: A label is presented to all users whereas the name may be hidden and only exposed by assistive technology. In many (but not all) cases the name and the label are the same.

Note 2: The term label is not limited to the label element in HTML.

large scale (text)

with at least 18 point or 14 point bold or font size that would yield equivalent size for Chinese, Japanese and Korean (CJK) fonts

Note 1: Fonts with extraordinarily thin strokes or unusual features and characteristics that reduce the familiarity of their letter forms are harder to read, especially at lower contrast levels.

Note 2: Font size is the size when the content is delivered. It does not include resizing that may be done by a user.

Note 3: The actual size of the character that a user sees is dependent both on the author-defined size and the user's display or user-agent settings. For many mainstream body text fonts, 14 and 18 point is roughly equivalent to 1.2 and 1.5 em or to 120% or 150% of the default size for body text (assuming that the body font is 100%), but authors would need to check this for the particular fonts in use. When fonts are defined in relative units, the actual point size is calculated by the user agent for display. The point size should be obtained from the user agent, or calculated based on font metrics as the user agent does, when evaluating this success criterion. Users who have low vision would be responsible for choosing appropriate settings.

Note 4: When using text without specifying the font size, the smallest font size used on major browsers for unspecified text would be a reasonable size to assume for the font. If a level 1 heading is rendered in 14pt bold or higher on major browsers, then it would be reasonable to assume it is large text. Relative scaling can be calculated from the default sizes in a similar fashion.

Note 5: The 18 and 14 point sizes for roman texts are taken from the minimum size for large print (14pt) and the larger standard font size (18pt). For other fonts such as CJK languages, the "equivalent" sizes would be the minimum large print size used for those languages and the next larger standard large print size.

legal commitments

transactions where the person incurs a legally binding obligation or benefit

Example: A marriage license, a stock trade (financial and legal), a will, a loan, adoption, signing up for the army, a contract of any type, etc.

link purpose

nature of the result obtained by activating a hyperlink

live

information captured from a real-world event and transmitted to the receiver with no more than a broadcast delay

Note 1: A broadcast delay is a short (usually automated) delay, for example used in order to give the broadcaster time to queue or censor the audio (or video) feed, but not sufficient to allow significant editing.

Note 2: If information is completely computer generated, it is not live.

lower secondary education level

the two or three year period of education that begins after completion of six years of school and ends nine years after the beginning of primary education

Note: This definition is based on the International Standard Classification of Education [UNESCO].

mechanism

process or technique for achieving a result

Note 1: The mechanism may be explicitly provided in the content, or may be relied upon to be provided by either the platform or by user agents, including assistive technologies.

Note 2: The mechanism needs to meet all success criteria for the conformance level claimed.

media alternative for text

media that presents no more information than is already presented in text (directly or via text alternatives)

Note: A media alternative for text is provided for those who benefit from alternate representations of text. Media alternatives for text may be audio-only, video-only (including sign-language video), or audio-video.

name
: text by which software can identify a component within Web content to the user

Note 1: The name may be hidden and only exposed by assistive technology, whereas a label is presented to all users. In many (but not all) cases, the label and the name are the same.

Note 2: This is unrelated to the name attribute in HTML.

navigated sequentially
: navigated in the order defined for advancing focus (from one element to the next) using a keyboard interface

non-text content
: any content that is not a sequence of characters that can be programmatically determined or where the sequence is not expressing something in human language

Note: This includes ASCII Art (which is a pattern of characters), emoticons, leetspeak (which uses character substitution), and images representing text

normative
: required for conformance

Note 1: One may conform in a variety of well-defined ways to this document.

Note 2: Content identified as "informative" or "non-normative" is never required for conformance.

on a full-screen window
: on the most common sized desktop/laptop display with the viewport maximized

Note: Since people generally keep their computers for several years, it is best not to rely on the latest desktop/laptop display resolutions but to consider the common desktop/laptop display resolutions over the course of several years when making this evaluation.

paused
: stopped by user request and not resumed until requested by user

prerecorded
: information that is not live

presentation
: rendering of the content in a form to be perceived by users

primary education level
: six year time period that begins between the ages of five and seven, possibly without any previous education

Note: This definition is based on the International Standard Classification of Education

[UNESCO].

process
: series of user actions where each action is required in order to complete an activity

Example 1: Successful use of a series of Web pages on a shopping site requires users to view alternative products, prices and offers, select products, submit an order, provide shipping information and provide payment information.

Example 2: An account registration page requires successful completion of a Turing test before the registration form can be accessed.

programmatically determined (programmatically determinable)
: determined by software from author-supplied data provided in a way that different user agents, including assistive technologies, can extract and present this information to users in different modalities

Example 1: Determined in a markup language from elements and attributes that are accessed directly by commonly available assistive technology.

Example 2: Determined from technology-specific data structures in a non-markup language and exposed to assistive technology via an accessibility API that is supported by commonly available assistive technology.

programmatically determined link context
: additional information that can be programmatically determined from relationships with a link, combined with the link text, and presented to users in different modalities

Example: In HTML, information that is programmatically determinable from a link in English includes text that is in the same paragraph, list, or table cell as the link or in a table header cell that is associated with the table cell that contains the link.

Note: Since screen readers interpret punctuation, they can also provide the context from the current sentence, when the focus is on a link in that sentence.

programmatically set
: set by software using methods that are supported by user agents, including assistive technologies

pure decoration
: serving only an aesthetic purpose, providing no information, and having no functionality

Note: Text is only purely decorative if the words can be rearranged or substituted without changing their purpose.

Example: The cover page of a dictionary has random words in very light text in the background.

real-time event
: event that a) occurs at the same time as the viewing and b) is not completely generated by the content

Example 1: A Webcast of a live performance (occurs at the same time as the viewing and is not prerecorded).

Example 2: An on-line auction with people bidding (occurs at the same time as the viewing).

Example 3: Live humans interacting in a virtual world using avatars (is not completely generated by the content and occurs at the same time as the viewing).

relationships
: meaningful associations between distinct pieces of content

relative luminance
: the relative brightness of any point in a colorspace, normalized to 0 for darkest black and 1 for lightest white

Note 1: For the sRGB colorspace, the relative luminance of a color is defined as L = 0.2126 * R + 0.7152 * G + 0.0722 * B where R, G and B are defined as:

- if R_{sRGB} <= 0.03928 then R = R_{sRGB}/12.92 else R = ((R_{sRGB}+0.055)/1.055) ^ 2.4
- if G_{sRGB} <= 0.03928 then G = G_{sRGB}/12.92 else G = ((G_{sRGB}+0.055)/1.055) ^ 2.4
- if B_{sRGB} <= 0.03928 then B = B_{sRGB}/12.92 else B = ((B_{sRGB}+0.055)/1.055) ^ 2.4

and R_{sRGB}, G_{sRGB}, and B_{sRGB} are defined as:

- R_{sRGB} = R_{8bit}/255
- G_{sRGB} = G_{8bit}/255
- B_{sRGB} = B_{8bit}/255

The "^" character is the exponentiation operator. (Formula taken from [sRGB] and [IEC-4WD]).

Note 2: Almost all systems used today to view Web content assume sRGB encoding. Unless it is known that another color space will be used to process and display the content, authors should evaluate using sRGB colorspace. If using other color spaces, see Understanding Success Criterion 1.4.3.

Note 3: If dithering occurs after delivery, then the source color value is used. For colors that are dithered at the source, the average values of the colors that are dithered should be used (average R, average G, and average B).

Note 4: Tools are available that automatically do the calculations when testing contrast and flash.

Note 5: A MathML version of the relative luminance definition is available.

relied upon (technologies that are)
: the content would not conform if that technology is turned off or is not supported

role
: text or number by which software can identify the function of a component within Web

content

Example: A number that indicates whether an image functions as a hyperlink, command button, or check box.

same functionality

same result when used

Example: A submit "search" button on one Web page and a "find" button on another Web page may both have a field to enter a term and list topics in the Web site related to the term submitted. In this case, they would have the same functionality but would not be labeled consistently.

same relative order

same position relative to other items

Note: Items are considered to be in the same relative order even if other items are inserted or removed from the original order. For example, expanding navigation menus may insert an additional level of detail or a secondary navigation section may be inserted into the reading order.

satisfies a success criterion

the success criterion does not evaluate to 'false' when applied to the page

section

A self-contained portion of written content that deals with one or more related topics or thoughts

Note: A section may consist of one or more paragraphs and include graphics, tables, lists and sub-sections.

set of Web pages

collection of Web pages that share a common purpose and that are created by the same author, group or organization

Note: Different language versions would be considered different sets of Web pages.

sign language

a language using combinations of movements of the hands and arms, facial expressions, or body positions to convey meaning

sign language interpretation

translation of one language, generally a spoken language, into a sign language

Note: True sign languages are independent languages that are unrelated to the spoken language(s) of the same country or region.

specific sensory experience

a sensory experience that is not purely decorative and does not primarily convey important information or perform a function

Example: Examples include a performance of a flute solo, works of visual art etc.

structure

1. The way the parts of a Web page are organized in relation to each other; and
2. The way a collection of Web pages is organized

supplemental content
: additional content that illustrates or clarifies the primary content

  Example 1: An audio version of a Web page.

  Example 2: An illustration of a complex process.

  Example 3: A paragraph summarizing the major outcomes and recommendations made in a research study.

synchronized media
: audio or video synchronized with another format for presenting information and/or with time-based interactive components, unless the media is a media alternative for text that is clearly labeled as such

technology (Web content)
: mechanism for encoding instructions to be rendered, played or executed by user agents

  Note 1: As used in these guidelines "Web Technology" and the word "technology" (when used alone) both refer to Web Content Technologies.

  Note 2: Web content technologies may include markup languages, data formats, or programming languages that authors may use alone or in combination to create end-user experiences that range from static Web pages to synchronized media presentations to dynamic Web applications.

  Example: Some common examples of Web content technologies include HTML, CSS, SVG, PNG, PDF, Flash, and JavaScript.

text
: sequence of characters that can be programmatically determined, where the sequence is expressing something in human language

text alternative
: Text that is programmatically associated with non-text content or referred to from text that is programmatically associated with non-text content. Programmatically associated text is text whose location can be programmatically determined from the non-text content.

  Example: An image of a chart is described in text in the paragraph after the chart. The short text alternative for the chart indicates that a description follows.

  Note: Refer to Understanding Text Alternatives for more information.

used in an unusual or restricted way
: words used in such a way that requires users to know exactly which definition to apply in order to understand the content correctly

  Example: The term "gig" means something different if it occurs in a discussion of music concerts than it does in article about computer hard drive space, but the appropriate definition can be determined from context. By contrast, the word "text" is used in a very specific way in WCAG 2.0, so a definition is supplied in the glossary.

user agent
: any software that retrieves and presents Web content for users

  Example: Web browsers, media players, plug-ins, and other programs — including assistive technologies — that help in retrieving, rendering, and interacting with Web content.

user-controllable
: data that is intended to be accessed by users

  Note: This does not refer to such things as Internet logs and search engine monitoring data.

  Example: Name and address fields for a user's account.

user interface component
: a part of the content that is perceived by users as a single control for a distinct function

  Note 1: Multiple user interface components may be implemented as a single programmatic element. Components here is not tied to programming techniques, but rather to what the user perceives as separate controls.

  Note 2: User interface components include form elements and links as well as components generated by scripts.

  Example: An applet has a "control" that can be used to move through content by line or page or random access. Since each of these would need to have a name and be settable independently, they would each be a "user interface component."

video
: the technology of moving or sequenced pictures or images

  Note: Video can be made up of animated or photographic images, or both.

video-only
: a time-based presentation that contains only video (no audio and no interaction)

viewport
: object in which the user agent presents content

  Note 1: The user agent presents content through one or more viewports. Viewports include windows, frames, loudspeakers, and virtual magnifying glasses. A viewport may contain another viewport (e.g., nested frames). Interface components created by the user agent such as prompts, menus, and alerts are not viewports.

  Note 2: This definition is based on User Agent Accessibility Guidelines 1.0 Glossary.

visually customized
: the font, size, color, and background can be set

Web page
: a non-embedded resource obtained from a single URI using HTTP plus any other

resources that are used in the rendering or intended to be rendered together with it by a user agent

Note 1: Although any "other resources" would be rendered together with the primary resource, they would not necessarily be rendered simultaneously with each other.

Note 2: For the purposes of conformance with these guidelines, a resource must be "non-embedded" within the scope of conformance to be considered a Web page.

Example 1: A Web resource including all embedded images and media.

Example 2: A Web mail program built using Asynchronous JavaScript and XML (AJAX). The program lives entirely at http://example.com/mail, but includes an inbox, a contacts area and a calendar. Links or buttons are provided that cause the inbox, contacts, or calendar to display, but do not change the URI of the page as a whole.

Example 3: A customizable portal site, where users can choose content to display from a set of different content modules.

Example 4: When you enter "http://shopping.example.com/" in your browser, you enter a movie-like interactive shopping environment where you visually move around in a store dragging products off of the shelves around you and into a visual shopping cart in front of you. Clicking on a product causes it to be demonstrated with a specification sheet floating alongside. This might be a single-page Web site or just one page within a Web site.

Appendix B: Acknowledgments

This section is informative.

This publication has been funded in part with Federal funds from the U.S. Department of Education, National Institute on Disability and Rehabilitation Research (NIDRR) under contract number ED05CO0039. The content of this publication does not necessarily reflect the views or policies of the U.S. Department of Education, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

Additional information about participation in the Web Content Accessibility Guidelines Working Group (WCAG WG) can be found on the Working Group home page.

Participants active in the WCAG WG at the time of publication

- Bruce Bailey (U.S. Access Board)
- Frederick Boland (NIST)
- Ben Caldwell (Trace R&D Center, University of Wisconsin)
- Sofia Celic (W3C Invited Expert)
- Michael Cooper (W3C)
- Roberto Ellero (International Webmasters Association / HTML Writers Guild)
- Bengt Farre (Rigab)

- Loretta Guarino Reid (Google)
- Katie Haritos-Shea
- Andrew Kirkpatrick (Adobe)
- Drew LaHart (IBM)
- Alex Li (SAP AG)
- David MacDonald (E-Ramp Inc.)
- Roberto Scano (International Webmasters Association / HTML Writers Guild)
- Cynthia Shelly (Microsoft)
- Andi Snow-Weaver (IBM)
- Christophe Strobbe (DocArch, K.U.Leuven)
- Gregg Vanderheiden (Trace R&D Center, University of Wisconsin)

Other previously active WCAG WG participants and other contributors to WCAG 2.0

Shadi Abou-Zahra, Jim Allan, Jenae Andershonis, Avi Arditti, Aries Arditi, Mike Barta, Sandy Bartell, Kynn Bartlett, Marco Bertoni, Harvey Bingham, Chris Blouch, Paul Bohman, Patrice Bourlon, Judy Brewer, Andy Brown, Dick Brown, Doyle Burnett, Raven Calais, Tomas Caspers, Roberto Castaldo, Sambhavi Chandrashekar, Mike Cherim, Jonathan Chetwynd, Wendy Chisholm, Alan Chuter, David M Clark, Joe Clark, James Coltham, James Craig, Tom Croucher, Nir Dagan, Daniel Dardailler, Geoff Deering, Pete DeVasto, Don Evans, Neal Ewers, Steve Faulkner, Lainey Feingold, Alan J. Flavell, Nikolaos Floratos, Kentarou Fukuda, Miguel Garcia, P.J. Gardner, Greg Gay, Becky Gibson, Al Gilman, Kerstin Goldsmith, Michael Grade, Jon Gunderson, Emmanuelle Gutiérrez y Restrepo, Brian Hardy, Eric Hansen, Sean Hayes, Shawn Henry, Hans Hillen, Donovan Hipke, Bjoern Hoehrmann, Chris Hofstader, Yvette Hoitink, Carlos Iglesias, Ian Jacobs, Phill Jenkins, Jyotsna Kaki, Leonard R. Kasday, Kazuhito Kidachi, Ken Kipness, Marja-Riitta Koivunen, Preety Kumar, Gez Lemon, Chuck Letourneau, Scott Luebking, Tim Lacy, Jim Ley, William Loughborough, Greg Lowney, Luca Mascaro, Liam McGee, Jens Meiert, Niqui Merret, Alessandro Miele, Mathew J Mirabella, Charles McCathieNevile , Matt May, Marti McCuller, Sorcha Moore, Charles F. Munat, Robert Neff, Bruno von Niman, Tim Noonan, Sebastiano Nutarelli, Graham Oliver, Sean B. Palmer, Sailesh Panchang, Nigel Peck, Anne Pemberton, David Poehlman, Adam Victor Reed, Chris Ridpath, Lee Roberts, Gregory J. Rosmaita, Matthew Ross, Sharron Rush, Gian Sampson-Wild, Joel Sanda, Gordon Schantz, Lisa Seeman, John Slatin, Becky Smith, Jared Smith, Neil Soiffer, Jeanne Spellman, Mike Squillace, Michael Stenitzer, Jim Thatcher, Terry Thompson, Justin Thorp, Makoto Ueki, Eric Velleman, Dena Wainwright, Paul Walsch, Takayuki Watanabe, Jason White.

Appendix C: References

This section is informative.

CAPTCHA

The CAPTCHA Project, Carnegie Mellon University. The project is online at http://www.captcha.net.

HARDING-BINNIE

Harding G. F. A. and Binnie, C.D., Independent Analysis of the ITC Photosensitive Epilepsy Calibration Test Tape. 2002.

IEC-4WD

IEC/4WD 61966-2-1: Colour Measurement and Management in Multimedia Systems and Equipment - Part 2.1: Default Colour Space - sRGB. May 5, 1998.

sRGB

"A Standard Default Color Space for the Internet - sRGB," M. Stokes, M. Anderson, S. Chandrasekar, R. Motta, eds., Version 1.10, November 5, 1996. A copy of this paper is available at http://www.w3.org/Graphics/Color/sRGB.html.

UNESCO

International Standard Classification of Education, 1997. A copy of the standard is available at http://www.unesco.org/education/information/nfsunesco/doc/isced_1997.htm.

WCAG10

Web Content Accessibility Guidelines 1.0, G. Vanderheiden, W. Chisholm, I. Jacobs, Editors, W3C Recommendation, 5 May 1999, http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/. The latest version of WCAG 1.0 is available at http://www.w3.org/TR/WAI-WEBCONTENT/.

Techniques for WCAG 2.0

PDF Techniques for WCAG 2.0

This Web page lists PDF Techniques from Techniques for WCAG 2.0: Techniques and Failures for Web Content Accessibility Guidelines 2.0. Technology-specific techniques do not replace the general techniques: content developers should consider both general techniques and technology-specific techniques as they work toward conformance.

Publication of techniques for a specific technology does not imply that the technology can be used in all situations to create content that meets WCAG 2.0 success criteria and conformance requirements. Developers need to be aware of the limitations of specific technologies and provide content in a way that is accessible to people with disabilities.

For information about the techniques, see Introduction to Techniques for WCAG 2.0. For a list of techniques for other technologies, see the Table of Contents.

Table of Contents

PDF8: Providing definitions for abbreviations via an E entry for a structure element

PDF9: Providing headings by marking content with heading tags in PDF documents

PDF10: Providing labels for interactive form controls in PDF documents

PDF11: Providing links and link text using the Link annotation and the /Link structure element in PDF documents

PDF12: Providing name, role, value information for form fields in PDF documents

PDF13: Providing replacement text using the /Alt entry for links in PDF documents

PDF14: Providing running headers and footers in PDF documents

PDF15: Providing submit buttons with the submit-form action in PDF forms

PDF16: Setting the default language using the /Lang entry in the document catalog of a PDF document

PDF17: Specifying consistent page numbering for PDF documents

PDF18: Specifying the document title using the Title entry in the document information dictionary of a PDF document

PDF19: Specifying the language for a passage or phrase with the Lang entry in PDF documents

PDF20: Using Adobe Acrobat Pro's Table Editor to repair mistagged tables

PDF21: Using List tags for lists in PDF documents

PDF22: Indicating when user input falls outside the required format or values in PDF forms

PDF23: Providing interactive form controls in PDF documents

✧ ✧ ✧

PDF Technology Notes

Introduction

The Portable Document Format (PDF) is a file format for representing documents in a manner independent of the application software, hardware, and operating system used to create them, as well as of the output device on which they are to be displayed or printed. PDF files specify the appearance of pages in a document in a reliable, device-independent manner. The PDF specification was introduced by Adobe Systems in 1993 as a publicly available standard. In July 2008, PDF 1.7 became an ISO standard (ISO 32000-1) [ISO32000].

Of note for accessibility is PDF/UA (Universal Accessibility) which became an ISO Standard in July 2012, and was updated in 2014 (ISO 14289-1:2014 (See PDF/UA (ISO 14289-1:2014).) The scope of PDF/UA is not meant to be a techniques (how-to) specification, but rather a set of guidelines for creating more accessible PDF. The specification describes the required and prohibited components and the conditions governing their inclusion in or exclusion from a PDF file in order for the file to be available to the widest possible audience, including those with disabilities. The mechanisms for including the components in the PDF stream are left to the discretion of the individual developer, PDF generator, or PDF viewing agent. PDF/UA also specifies the rules governing the behavior for a conforming reader.

PDF Accessibility Support

PDF includes several features in support of accessibility of documents to users with disabilities. The core of this support lies in the ability to determine the logical order of content in a PDF document, independently of the content's appearance or layout, through logical structure and Tagged PDF. Applications can extract the content of a document for presentation to users with disabilities by traversing the structure hierarchy and presenting the contents of each node. For this reason, producers of PDF files must ensure that all information in a document is reachable by means of the structure hierarchy.

Logical Structure

PDF's logical structure features (introduced in PDF 1.3) provide a mechanism for incorporating structural information about a document's content into a PDF file. Such information might include, for example, the organization of the document into chapters, headings, paragraphs and sections or the identification of special elements such as figures, tables, and footnotes. The logical structure features are extensible, allowing applications that produce PDF files to choose what structural information to include and how to represent it, while enabling PDF consumers to navigate a file without knowing the producer's structural conventions.

PDF logical structure shares basic features with standard document markup languages such as HTML, SGML, and XML. A document's logical structure is expressed as a hierarchy of structure elements, each represented by a dictionary object. Like their counterparts in other markup languages, PDF structure elements can have content and attributes. In PDF, rendered document content takes over the role occupied by text in HTML, SGML, and XML.

A PDF document's logical structure is stored separately from its visible content, with pointers from each to the other. This separation allows the ordering and nesting of logical elements to be entirely independent of the order and location of graphics objects on the document's pages.

The logical structure of a document is described by a hierarchy of objects called the structure hierarchy or structure tree. At the root of the hierarchy is a dictionary object called the structure tree root, located by means of the StructTreeRoot entry in the document catalog. See Section 14.7.2, ("Structure Hierarchy") in PDF 1.7 (ISO 32000-1): Table 322 shows the entries in the structure tree root dictionary. The K entry specifies the immediate children of the structure tree root, which are structure elements.

Tagged PDF

Tagged PDF (PDF 1.4) is a stylized use of PDF that builds on PDF's logical structure framework. It defines a set of standard structure types and attributes that allow page content (text, graphics, and images) to be extracted and reused for other purposes. It is intended for use by tools that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
- Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible to people who rely on assistive technology.

PDF File Production and Accessibility

PDF files may be produced either directly by application programs or indirectly by conversion from other file formats or imaging models. In addition, tools exist for inspecting, checking, and repairing

PDF files for accessibility. The following sections provide representative lists of applications and tools typically used for these functions.

These notes do not, and cannot, provide an exhaustive list, nor do they endorse particular applications and tools. Rather they provide a snapshot of tools in fairly wide use at the time the WCAG Working Group undertook to review and publish techniques for producing PDF documents. As with any software, application support for PDF accessibility will vary with different versions, with the formatting requirements of specific PDF documents, and with actual usage of the application. That is, the tools can be used properly to produce appropriate tags, etc..

In general, newer tools will provide greater support than earlier ones. The tools' vendors are the source of authoritative information about their support for PDF accessibility.

Generating PDF Files

Many applications can generate PDF files directly, and some can import them as well. This direct approach is preferable, since it gives the application access to the full capabilities of PDF, including the imaging model and the interactive and document interchange features. Alternatively, applications that do not generate PDF directly can produce PDF output indirectly. There are two principal indirect methods:

- The application describes its printable output by making calls to an application programming interface (API) such as GDI in Microsoft® Windows® or QuickDraw in the Apple Mac OS. A software component called a printer driver intercepts these calls and interprets them to generate output in PDF form.
- The application produces printable output directly in some other file format, such as PostScript, PCL, HPGL, or DVI, which is converted to PDF by a separate translation program.

Although these indirect strategies are often the easiest way to obtain PDF output from an existing application, the resulting PDF files may not make the best use of the high-level PDF imaging model relied upon to expose the semantics of the document. This is because the information embodied in the application's API calls or in the intermediate output file often describes the desired results at too low a level. Any higher-level information maintained by the original application has been lost and is not available to the printer driver or translator.

For example, since the printer driver or translator targets correct visual output, information about the semantics of the document may not be sent at all or may be ignored when creating the PDF output. As a result, headings may not be tagged as such, or link text may not be associated with its link object. Check with the vendor of any PDF authoring tool in order to understand how to use the tool in a way that produces the best tagged output.

PDF Authoring Tools that Provide Accessibility Support

- Adobe Acrobat's PDFMaker - PDFMaker is part of Adobe Acrobat which adds macros to many business applications such as Microsoft Office, AutoCAD and Lotus Notes that support the conversion of content from the original format to tagged PDF.
- Adobe FrameMaker - Desktop publishing application from Adobe Systems that directly exports tagged PDF and provides support for alternative text descriptions.
- Adobe InDesign - Page layout and desktop publishing application from Adobe Systems that directly exports tagged PDF and provides support for alternative text descriptions.
- Adobe LiveCycle Designer - Windows-based forms design application from Adobe Systems that directly exports tagged PDF interactive forms and provides support for alternative text descriptions; can be invoked standalone or from within Acrobat Pro.
- LibreOffice - Open-source word processing software from The Document Foundation that can export tagged PDF.
- Lotus Symphony Documents - Word-processing software from IBM that can export tagged PDF.
- Microsoft® Word - Word processing application from Microsoft Corporation that can export tagged PDF using the save as XPS or PDF utility.
- OpenOffice.org Writer - Open source word-processing software from Sun Microsystems Inc. that can export tagged PDF using the Export as PDF utility.
- CommonLook Office for Microsoft Office from Netcentric Technologies is an add-in to Microsoft® Word and PowerPoint that makes it possible to create tagged PDF documents. CommonLook Office provides tools to allow content authors to run accessibility tests in the Microsoft Word and PowerPoint environments and to remediate accessibility issues prior to conversion to PDF.
- Xenos Axess™ for Accessible Statements - PDF software integrates with an organization's existing enterprise content management (ECM) infrastructure to capture high-volume print streams and automatically transform them into tagged PDFs.
- WordPerfect® Office - Word-processing software from Corel that can be used to create, mark up, and share tagged PDF documents.
- Microsoft Office 10 - a suite of desktop office applications that creates tagged PDF.

Note: Care should be taken when choosing PDF creation tools from the many available, as some may not support creation of tagged PDF files.

Accessibility Checking and Repair

Adobe Acrobat Pro. Adobe Acrobat Pro is an application that creates and edits PDF files. It has a number of tools for evaluating and repairing the accessibility of PDF files, including access to the structure root through the tags panel, the ability to directly manipulate the reading order through the order panel, a built-in accessibility checker, and the Touch Up Reading Order tool which provides a graphical mechanism for assessing and repairing the accessibility of a PDF document.

Commonlook™ PDF. Commonlook PDF. Commonlook PDF is a plug-in for Adobe Acrobat Pro from Netcentric Technologies. CommonLook PDF helps identify, report and correct the most common accessibility problems, including the proper tagging of images, tables, forms and other non-textual objects.

API Inspection Tools

- aDesigner - a disability simulator from the Eclipse Foundation that helps designers ensure that content is accessible and usable by visually impaired users.
- inspect32 - part of the Microsoft Windows Software Development Kit (SDK) that allows developers and testers to examine the accessible properties of UI components.
- PDDOMView - part of Acrobat_Accessibility_9.1.zip which contains files that can be used by Windows clients of the accessibility interfaces described in the Accessibility API Reference document.
- UISpy - part of the Microsoft Windows Software Development Kit (SDK) that allows developers and testers to view and interact with the user interface (UI) elements of an application.

User Agents

PDF User Agents with accessibility support include:

- Adobe Acrobat Pro - PDF Authoring Tool, Editor, and Viewer from Adobe Systems which is compatible with MSAA devices on the Windows platform. Has a number of built in accessibility features including text to speech (Read Out Loud), high contrast display, reflow for large print display, auto scroll, accessibility full check, accessibility quick check, touch up reading order tool, and an accessibility setup assistant.
- Adobe Reader – Freely distributed PDF Viewer from Adobe Systems which is compatible with MSAA devices on the Windows platform. Has a number of built in accessibility features including text to speech (Read Out Loud), high contrast display, reflow for large print display, auto scroll, accessibility quick check, and an accessibility setup assistant.
- Kurzweil 3000™ - a comprehensive reading, writing and learning software solution from Kurzweil Educational Systems® which reads PDF files using text to speech facilities.

Adobe Reader and Acrobat Support for Accessibility APIs

Adobe provides methods to make the content of a PDF file available to assistive technology such as screen readers:

- On the Microsoft® Windows® operating system, Acrobat and Adobe Reader export PDF content as Component Object Model (COM) objects. Accessibility applications such as screen readers can interface with Acrobat or Adobe Reader in two ways:
  - Through the Microsoft Active Accessibility (MSAA) interface, using MSAA objects that Acrobat or Adobe Reader exports
  - Directly through exported COM objects that allow access to the PDF document's internal structure, called the Document Object Model (DOM).
- On UNIX® platforms, Adobe Reader supports the Gnome accessibility architecture. C-based Accessibility Toolkit (ATK) interfaces are available.

The DOM and MSAA models are related, and developers can use either or both. Acrobat issues notifications to accessibility clients about interesting events occurring in the PDF file window and responds to requests from such clients. Recent versions of Acrobat and Reader have enhanced the support for accessibility interfaces:

- MSAA interfaces are supported in Acrobat/Reader 5.0 and later.
- In Acrobat/Reader 6.0 and later, information about the underlying PDF structure is made available through direct COM objects that represent the PDF DOM. The DOM accessibility interfaces provide somewhat more extensive access.
- In Acrobat/Reader 7.0 and later, ATK and expanded DOM support is available.
- The Linux®, Solaris™, AIX®, and HP-UX versions of Adobe Reader implement C-based ATK interfaces, allowing screen readers, screen magnifiers, and on-screen keyboards to query an Accessibility Technology - Service Providers Interface (AT-SPI) registry for applications that are accessible.
- The DOM has been expanded to provide enhanced caret, selection, and focus support, as well as the new interfaces IPDDomDocument, ISelectText, and IPDDomNodeExt.

Assistive Technology Support

- JAWS 12 for Windows - screen reader from Freedom Scientific. Support for PDF started with JAWS version 4.
- MAGic 11 - screen magnifier from Freedom Scientific
- NVDA 2011.1 - NonVisual Desktop Access, open source screen reader distributed by NV Access. Providing feedback via synthetic speech and Braille, NVDA allows blind and vision-impaired people to access and interact with the Windows operating system and many third party applications.

- Supernova Access Suite 12.02 – full screen reader offering magnification, speech, and Braille support from Dolphin. Support for PDF started with HAL version 5.
- System Access To Go - screen reader from Serotek Corporation
- VoiceOver - screen reader for Mac OS X v10.6 Snow Leopard
- Window-Eyes 7.2 - screen reader from GW Micro. Window-Eyes was the first screen reader to provide support for PDF files, in Window-Eyes 4.2.
- ZoomText 9.1 - screen magnifier and screen reader from Ai Squared, with support for Adobe Acrobat and Reader:
    - PDF documents can be read using both AppReader and DocReader (without special settings)
    - PDF documents can be read in all Windows operating systems (without special settings)
    - AppReader and DocReader start instantly in Adobe Reader
    - PDF documents can be read with greater accuracy and without paging delays
    - PDF documents can be read in Internet Explorer (with the Adobe Reader plug-in)
    - Special Adobe Reader settings are no longer needed to obtain optimal reading

Related References

- Adobe Accessibility Resource Center
- Adobe Acrobat Accessibility Training Resources
- Accessing PDF Documents with Assistive Technology
- PDF Specification Archives
- PDF 1.7 Reference: ISO approved copy of the ISO 32000-1
- PDF Accessibility API Reference - How AT developers can use Acrobat MSAA and IPDDom interfaces to provide access to PDF content
- PDF/UA (ISO 14289-1:2012)
- PDF/UA Conformance Testing Model: The Matterhorn Protocol
- WebAIM PDF Accessibility
- Create accessible PDFs using Microsoft Office 10

PDF1: Applying text alternatives to images with the Alt entry in PDF documents

Applicability

Tagged PDF documents with images

This technique relates to:

- Success Criterion 1.1.1 (Non-text Content)
  - How to Meet 1.1.1 (Non-text Content)
  - Understanding Success Criterion 1.1.1 (Non-text Content)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF1. Also see PDF Technology Notes.

Description

The objective of this technique is to provide text alternatives for images via an /Alt entry in the property list for a Tag. This is normally accomplished using a tool for authoring PDF.

PDF documents may be enhanced by providing alternative descriptions for images, formulas, and other items that do not translate naturally into text. In fact, such text alternatives are required for accessibility: alternate descriptions are human-readable text that can be vocalized by text-to-speech technology for the benefit of users with vision disabilities.

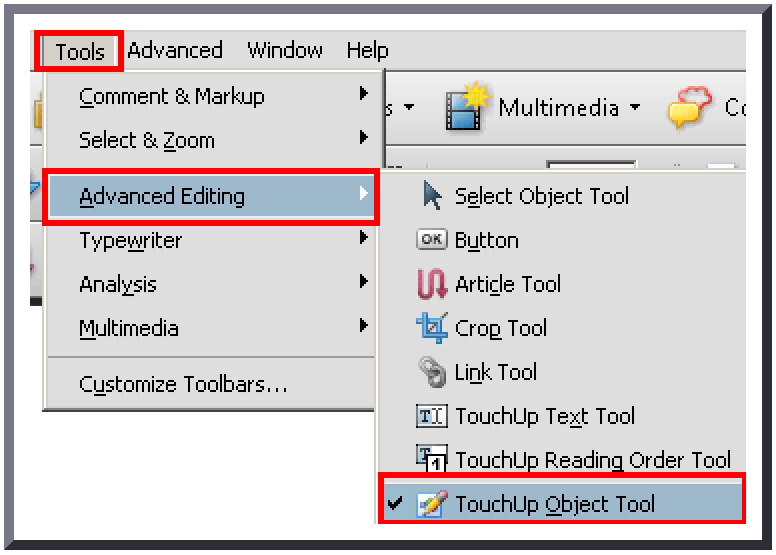
When an image contains words that are important to understanding the content, the text alternative should include those words. This will allow the alternative to accurately represent the image. Note that it does not necessarily describe the visual characteristics of the image itself but must convey the same meaning as the image.

Examples

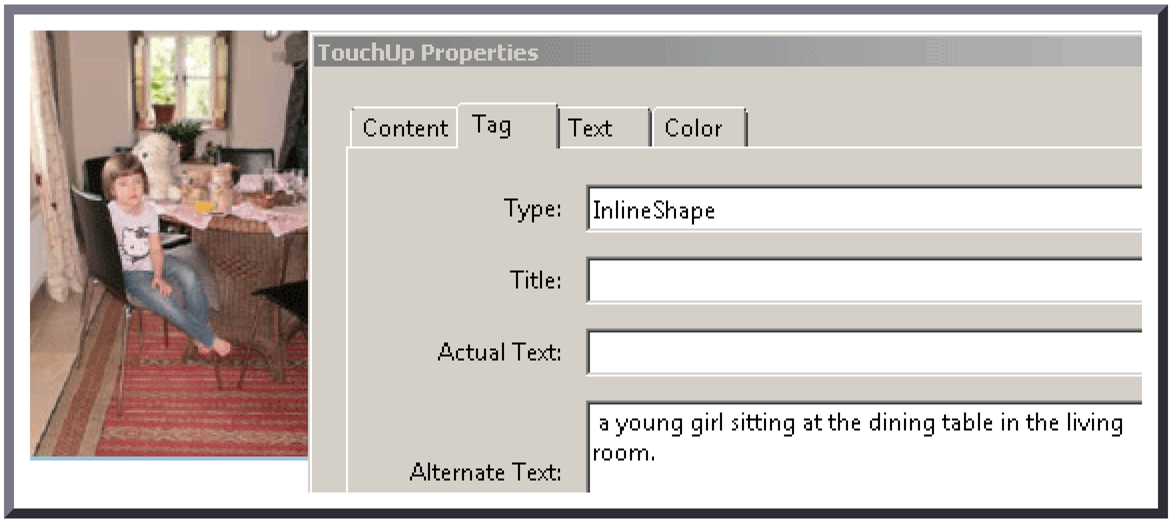
Example 1: Adding an /Alt entry to an image using Adobe Acrobat 9 Pro's TouchUp Object Tool

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Choose Tools > Advanced Editing > TouchUp Object Tool



2. Access the context menu for the image and choose Properties.
3. On the TouchUp Properties dialog, select the Tag tab.
4. On the Tag panel, type the text alternative in the Alternate Text text box.



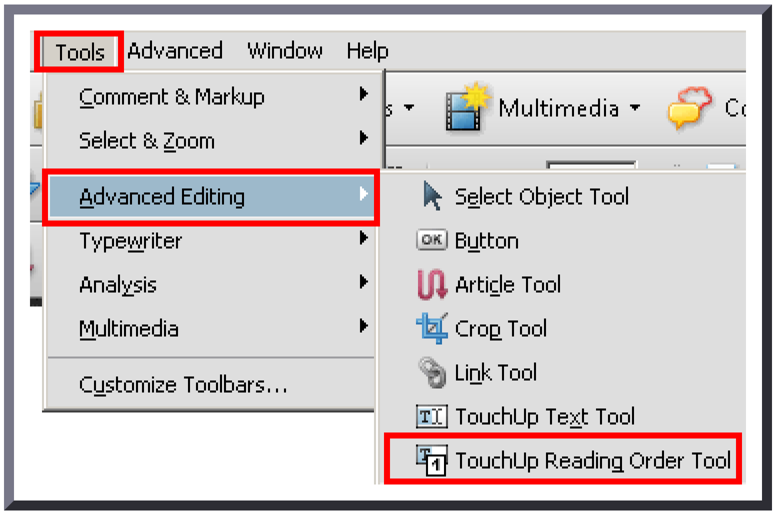
Type: InlineShape
Title:
Actual Text:
Alternate Text: a young girl sitting at the dining table in the living room.

This example is shown in operation in the working example of Adding an /Alt entry to an image.

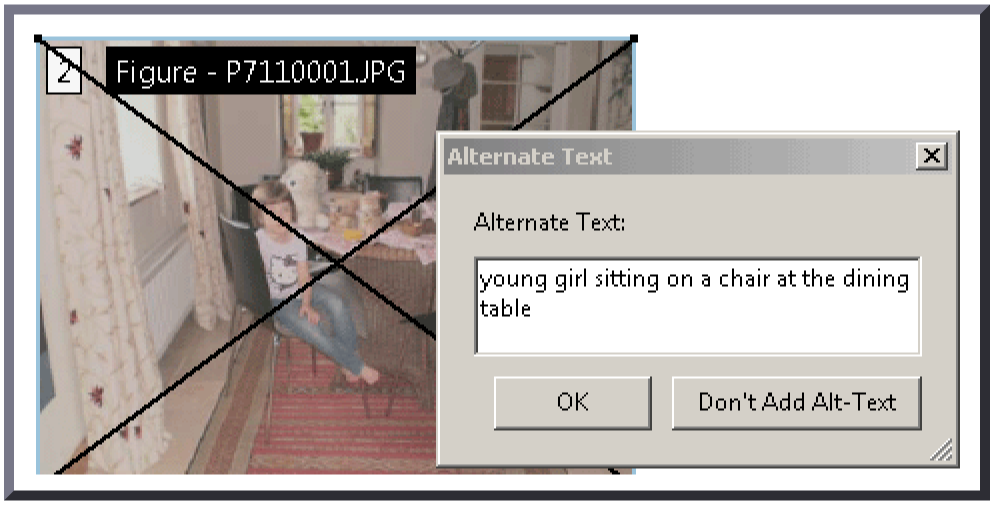
Example 2: Adding an /Alt entry to an image using Adobe Acrobat 9 Pro's TouchUp Reading Order Tool

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Choose Tools > Advanced Editing > TouchUp Reading Order Tool



2. The TouchUp Reading Order dialog will be displayed.
3. Right-click on the image and choose Edit Alternate Text.
4. The Alternate Text dialog will be displayed.
5. Type the text alternative in the Alternate Text text box.



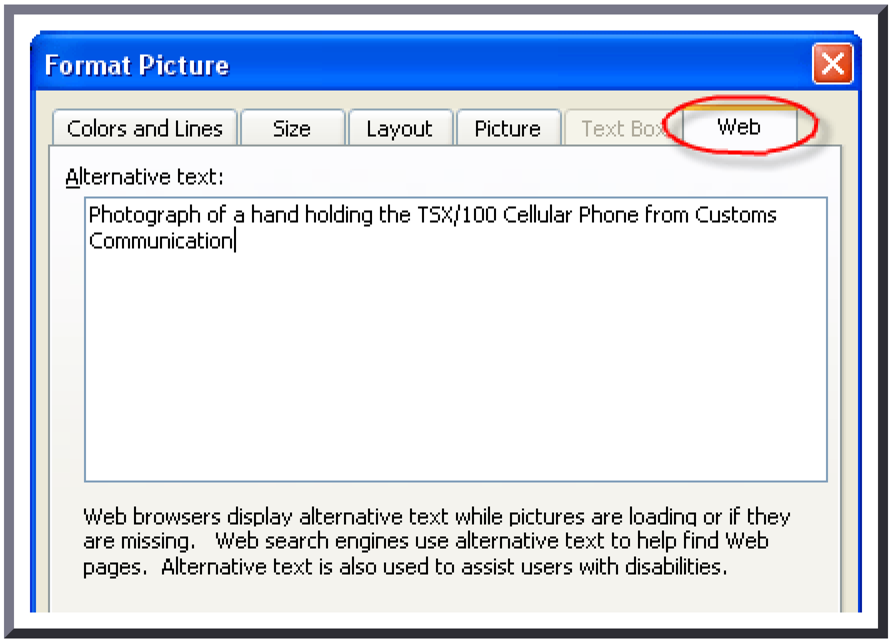
Alternate Text:
young girl sitting on a chair at the dining table

Example 3: Adding an /Alt entry to an image in PDF documents generated using Microsoft Word

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Word 2000-2003

1. Right-click on the image and choose Format Picture
2. Select the Web tab
3. Type the alternative text into the text box provided and then click OK.



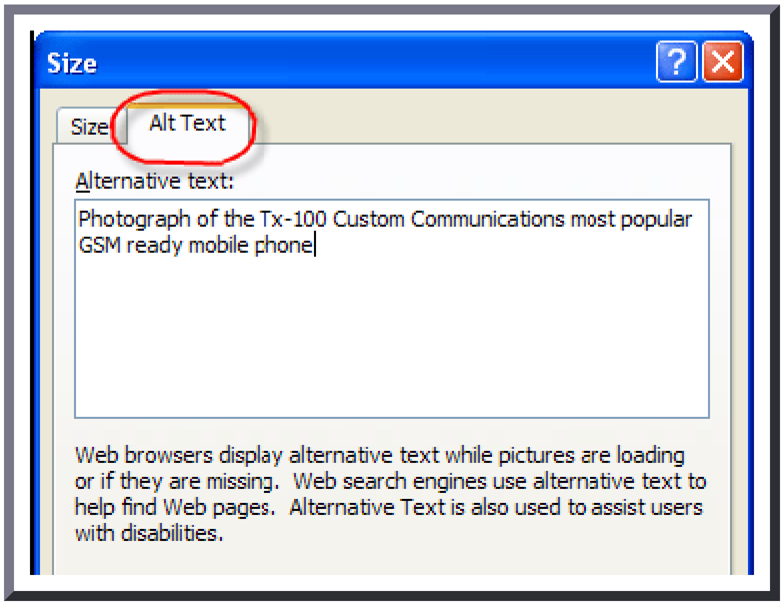
Alternative text:

Photograph of a hand holding the TSX/100 Cellular Phone from Customs Communication

Web browsers display alternative text while pictures are loading or if they are missing. Web search engines use alternative text to help find Web pages. Alternative text is also used to assist users with disabilities.

Word 2007

1. Right-click on the image and choose Size
2. Select the Alt Text tab
3. Type the alternative text into the text box provided and then click OK.



Alternative text:

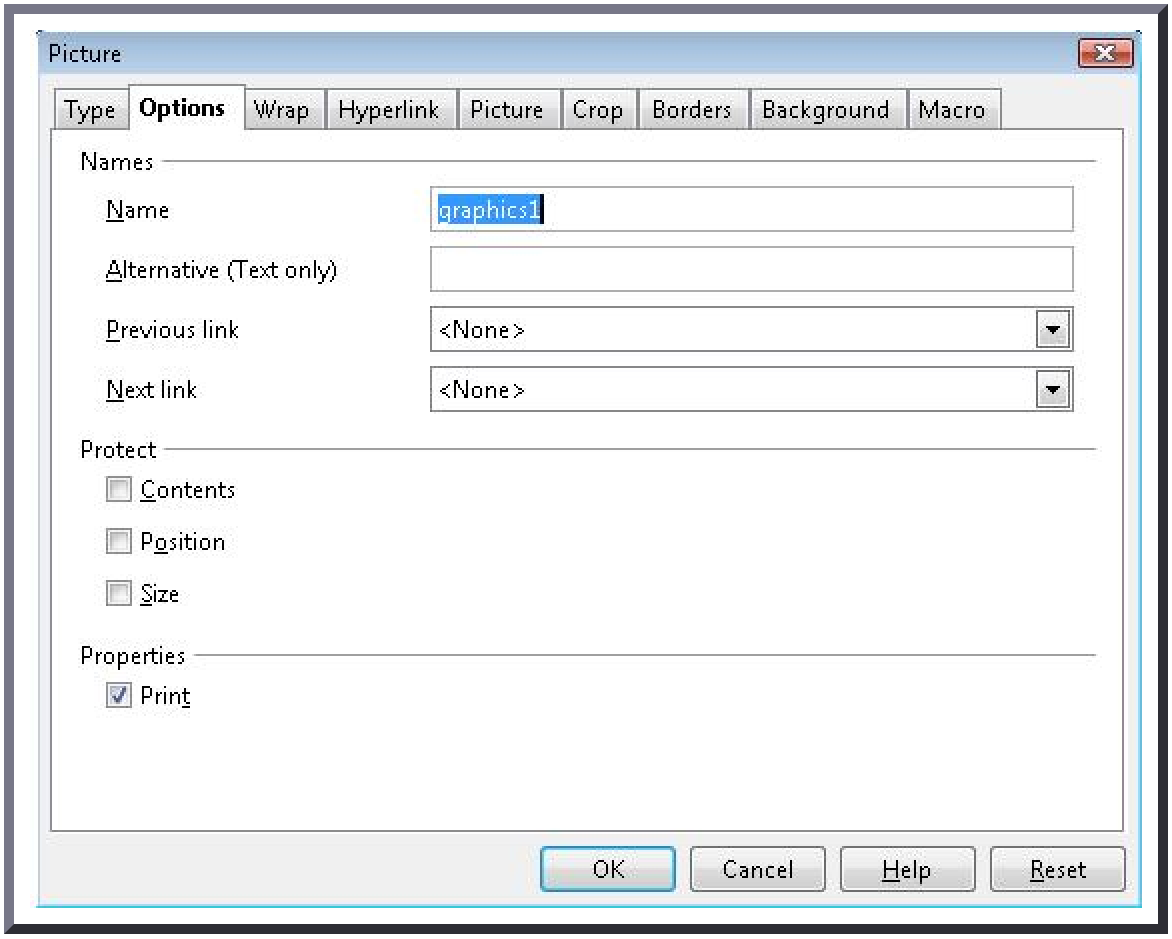
Photograph of the Tx-100 Custom Communications most popular GSM ready mobile phone

Web browsers display alternative text while pictures are loading or if they are missing. Web search engines use alternative text to help find Web pages. Alternative Text is also used to assist users with disabilities.

Example 4: Adding an /Alt entry to an image in PDF documents generated using OpenOffice.org Writer 2.2

This example is shown with Open Office.org Writer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Access the context menu for the image and choose Picture...
2. Select the Options tab
3. Type the alternative text into Alternate (Text Only) text box and click OK.



Names
Name: graphics1
Alternative (Text only):
Previous link: <None>
Next link: <None>
Protect
☐ Contents
☐ Position
☐ Size
Properties
☑ Print

Example 5: Adding a text alternative to an image in a PDF document using an /Alt entry

The /Alt property used on an image of mountains with a moon and trees typically would be used like this (typically accomplished by an authoring tool):

```
/Figure <</Alt (Sketch of Mountains with moon rising over trees) >>
```

The image might also be represented by a tag with a different name. A different name might be used because the tag name is written in a language other than English or because a specific tool uses a different name for some other reason. In this situation, it is also necessary that the RoleMap contained within the StructTreeRoot for the PDF document contain an entry which explicitly maps the name of the tag used for the image with the standard structure type used in PDF documents (in this case, Figure). If the RoleMap contains only an entry mapping Shape tags to Figure tags, the rolemap information would appear as follows:

```
/RoleMap << /Shape /Figure >>
```

In this case, the usage of the /Alt entry as follows would also be correct:

```
/Shape <</Alt (Crater Lake in the summer, with the blue sky, clouds and crater walls perfect
```

Note that the /Alt entry in property lists can be combined with other entries.

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.9.4 (Replacement Text) in PDF 1.7 (ISO 32000-1)
- Acrobat and Accessibility
- PDF Reference 1.6, 10.8.2 Alternate Descriptions
- PDF and Accessibility

Related Techniques

- G94: Providing short text alternative for non-text content that serves the same purpose and presents the same information as the non-text content

Tests

Procedure

1. Verify the images which need equivalents have /Alt entries on an enclosing tag by one of the following:
    - Read the PDF document with a screen reader, listening to hear that the equivalent text is read when tabbing to the non-text object (if it is tabbable) or hearing the alternative text read when reading the content line-by-line.
    - Using a PDF editor, check that a text alternative is displayed for each image.
    - Use a tool which is capable of showing the /Alt entry value, such as aDesigner, to open the PDF document and view the GUI summary to read the text alternatives for images.
    - Use a tool that exposes the document through the accessibility API and verify that images have required text equivalents.

Expected Results

- Check 1 is true for each image in the document which needs a text equivalent.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF2: Creating bookmarks in PDF documents

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 2.4.5 (Multiple Ways)
    - How to Meet 2.4.5 (Multiple Ways)
    - Understanding Success Criterion 2.4.5 (Multiple Ways)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF2. Also see PDF Technology Notes.

Description

The intent of this technique is to make it possible for users to locate content using bookmarks (*outline entries* in an Outline dictionary) in long documents.

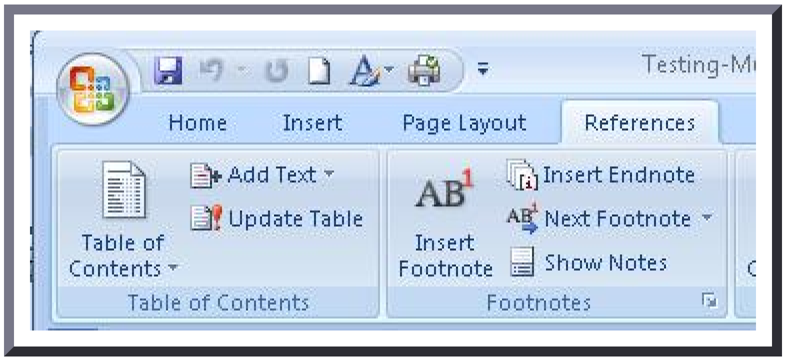
A person with cognitive disabilities may prefer a hierarchical outline that provides an overview of the document rather than reading and traversing through many pages. This is also a conventional means of navigating a document that benefits all users.

Examples

Example 1: Converting a table of contents created with Microsoft Word 2007 and creating bookmarks for Adobe Reader 9 and Acrobat 9 Pro

This example is shown with Microsoft Word and Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

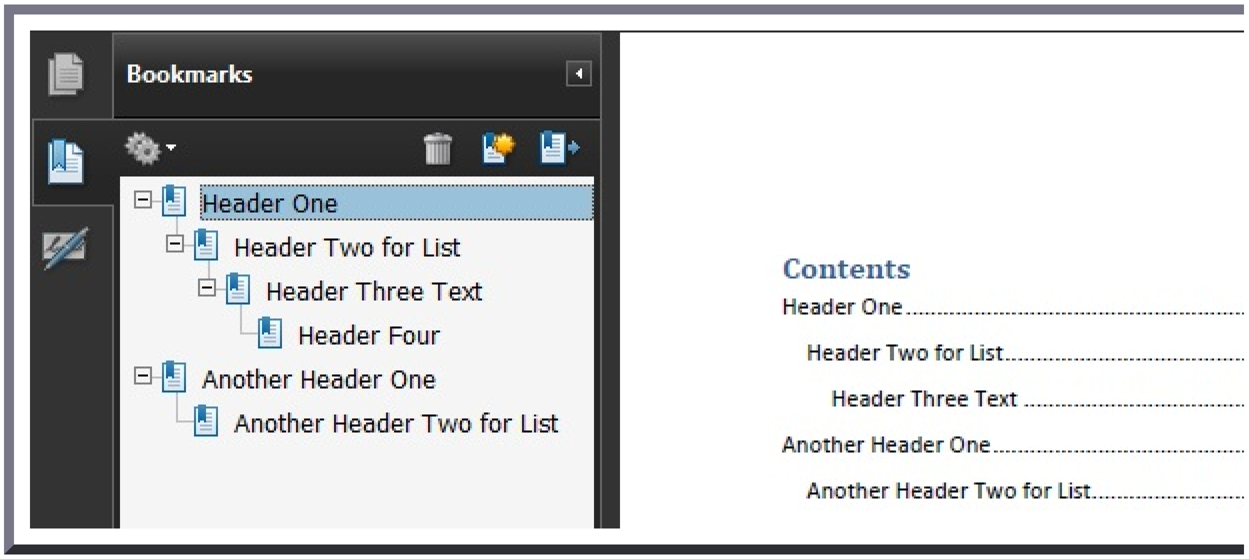
1. Create a table of contents at the beginning of the Word document.



2. Use Save as... > Adobe PDF to convert the Word document to PDF, specifying both of the following:
   - Enable Accessibility and Reflow with Tagged Adobe PDF
   - Convert Word Headings into Bookmarks

The table-of-contents entries in the converted document will be linked to the headings in the document.

In addition, the headings will appear as PDF Bookmarks in the left-hand Navigation pane.



If the document provides a glossary and/or index, these sections should have headings that appear in the table of contents (and thus as bookmarks in the Navigation pane). The table of contents also should be marked up with a heading so it is bookmarked as well.

If this markup has not been done in the authoring tool, Adobe Acrobat Pro can be used to provide the tags. See *PDF9: Providing headings by marking content with heading tags in PDF documents* if you need to modify converted headings or add new ones.

This example is shown in operation in the working example of creating bookmarks with Word 2007.

Example 2: Converting a table of contents created with OpenOffice.org Writer 2.2 and creating bookmarks for Adobe Reader 9 and Acrobat 9 Pro

This example is shown with OpenOffice.org Writer and Adobe Acrobat Pro and Reader. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Create a table of contents at the beginning of the OpenOffice.org Writer document:
   - Insert > Indexes and Tables... > Indexes and Tables > Insert Index/Table
2. Use File > Export as PDF... to convert the document to PDF, specifying Tagged PDF in the Options dialog.

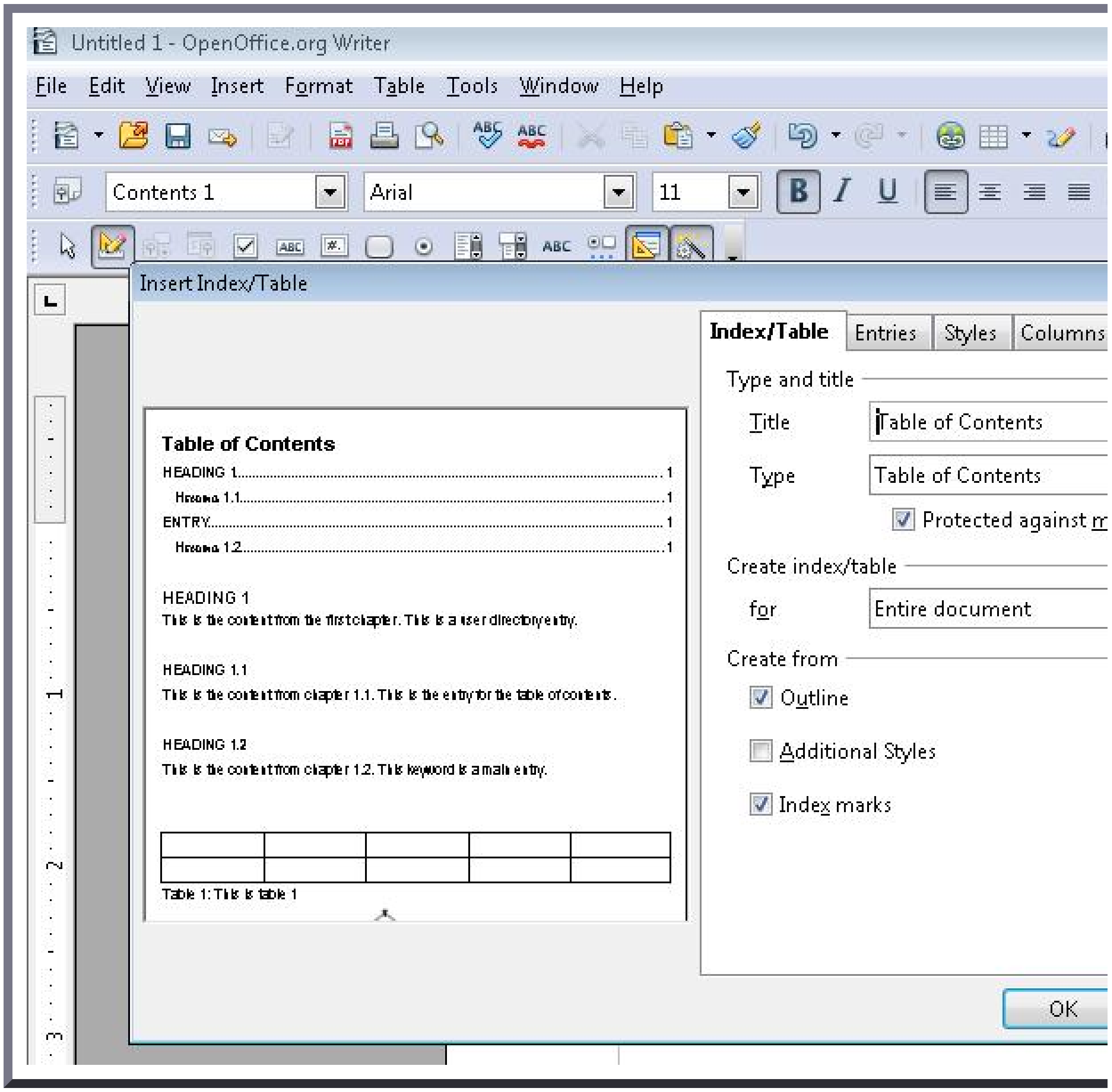

Untitled 1 - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Contents 1
Arial
11
Insert Index/Table
Index/Table
Entries
Styles
Columns
OK

Table of Contents
HEADING 1 1
Heading 1.1 1
ENTRY 1
Heading 1.2 1

HEADING 1
This is the content from the first chapter. This is a user directory entry.

HEADING 1.1
This is the content from chapter 1.1. This is the entry for the table of contents.

HEADING 1.2
This is the content from chapter 1.2. This keyword is a main entry.

Table 1: This is table 1

Type and title
Title: Table of Contents
Type: Table of Contents
[✓] Protected against m

Create index/table
for: Entire document

Create from
[✓] Outline
[ ] Additional Styles
[✓] Index marks

The table-of-contents entries in the converted document will be linked to the headings in the document, and will appear as PDF Bookmarks in the left-hand Navigation pane. The

OpenOffice.org Table of Contents and Bookmarks look the same as they appeared in Example 1.

This example is shown in operation in the working example of creating bookmarks with OpenOffice Writer.

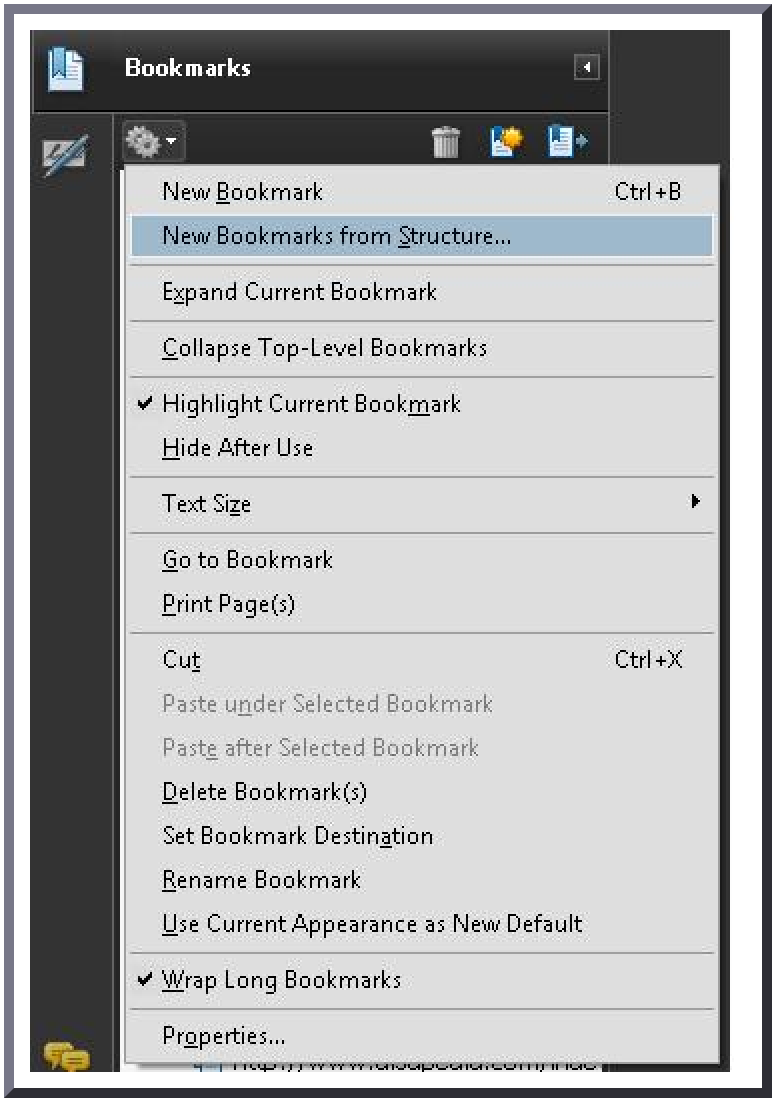
Example 3: Adding bookmarks using Adobe Acrobat 9 Pro after conversion

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

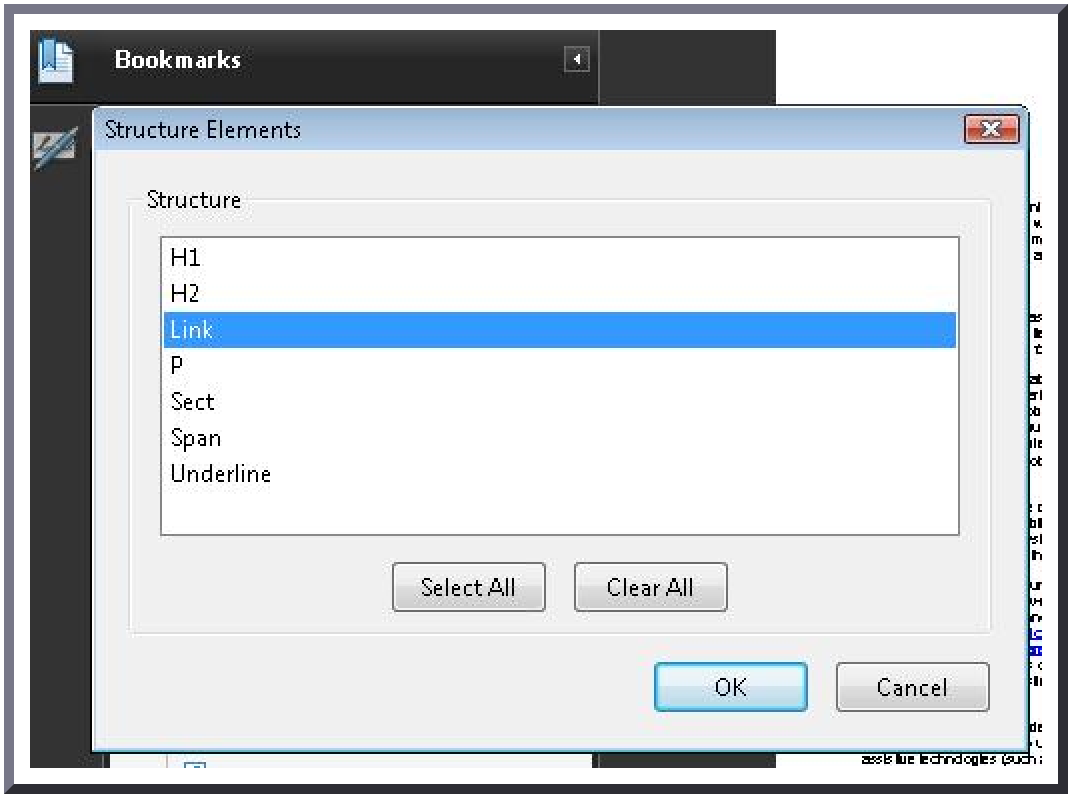
After conversion to tagged PDF, you may decide to add bookmarks that were not automatically generated. Like the converted bookmarks, tagged bookmarks use the underlying structural information in the document.

1. In the Bookmarks panel, choose the options menu, then choose New Bookmarks From Structure...
2. From the Structure Elements dialog, select the elements you want specified as tagged bookmarks.

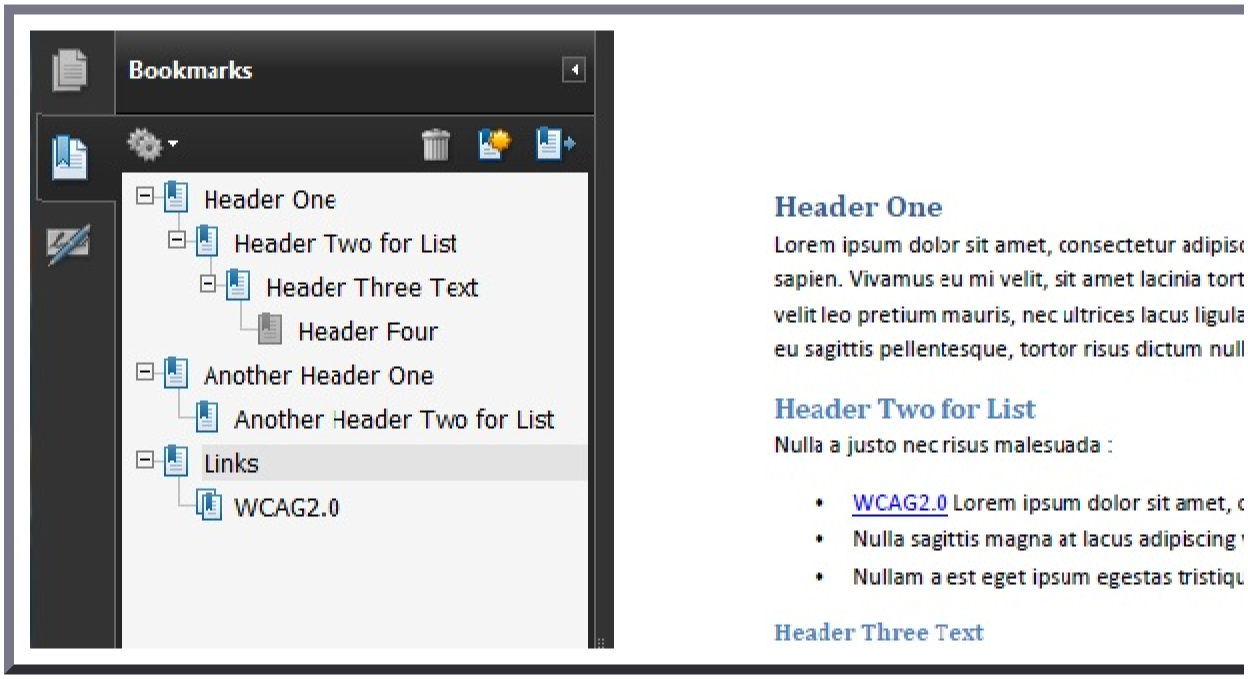
The image below shows the Bookmarks options menu.



The next image shows the selection of links in the document for bookmarking.



The tagged bookmarks are nested under a new, untitled bookmark. Access the context menu for the new bookmark and select the Rename option to rename the new bookmark, as shown in the following image.



Header One

Lorem ipsum dolor sit amet, consectetur adipis
sapien. Vivamus eu mi velit, sit amet lacinia tort
velit leo pretium mauris, nec ultrices lacus ligula
eu sagittis pellentesque, tortor risus dictum null

Header Two for List

Nulla a justo nec risus malesuada :

- WCAG2.0 Lorem ipsum dolor sit amet, c
- Nulla sagittis magna at lacus adipiscing
- Nullam a est eget ipsum egestas tristiqu

Header Three Text

This example is shown in operation in the [working example of creating bookmarks with Acrobat Pro](#).

Example 4: Creating bookmarks with the outline hierarchy

The following code fragment illustrates part of an outline hierarchy used to create bookmarks This is typically accomplished by an authoring tool.

```
121 0 obj
 << /Type /Outlines
    /First 22 0 R
    /Last 29 0 R
    /Count 6
 >>
endobj
22 0 obj
 << /Title (Applying Guerrilla Tactics to Usability Testing by People with Disabilities)
    /Parent 21 0 R
    /Next 29 0 R
    /First 25 0 R
    /Last 28 0 R
    /Count 4
    /Dest [3 0 R /XYZ 0 792 0]
 >>
endobj
25 0 obj
 << /Title (Getting started)
    /Parent 22 0 R
    /Next 26 0 R
    /Dest [3 0 R /XYZ null 701 null]
 >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 12.3.3 (Document Outline) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Related Techniques

- G64: Providing a Table of Contents

Tests

Procedure

1. Check that the Bookmarks panel displays bookmarks.
2. Check that the bookmarks link to the correct sections in the document.

Expected Results

- Check #1 and Check #2 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF3: Ensuring correct tab and reading order in PDF documents

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 1.3.2 (Meaningful Sequence)
  - How to Meet 1.3.2 (Meaningful Sequence)
  - Understanding Success Criterion 1.3.2 (Meaningful Sequence)
- Success Criterion 2.1.1 (Keyboard)
  - How to Meet 2.1.1 (Keyboard)
  - Understanding Success Criterion 2.1.1 (Keyboard)
- Success Criterion 2.1.3 (Keyboard (No Exception))
  - How to Meet 2.1.3 (Keyboard (No Exception))
  - Understanding Success Criterion 2.1.3 (Keyboard (No Exception))
- Success Criterion 2.4.3 (Focus Order)

- How to Meet 2.4.3 (Focus Order)
- Understanding Success Criterion 2.4.3 (Focus Order)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF3. Also see PDF Technology Notes.

Description

The intent of this technique is to ensure that users can navigate through content in a logical order that is consistent with the meaning of the content. Correct tab and reading order is typically accomplished using a tool for authoring PDF.

For sighted users, the logical order of PDF content is also the visual order on the screen. For keyboard and assistive technology users, the tab order through content, including interactive elements (form fields and links), determines the order in which these users can navigate the content. The tab order must reflect the logical order of the document.

Logical structure is created when a document is saved as tagged PDF. The reading order of a PDF document is determined primarily by the tag order of document elements, including interactive elements, but the order of content within individual tags is determined by the PDF document's content tree structure.

If the reading order is not correct, keyboard and assistive technology users may not be able to understand the content. For example, some documents use multiple columns, and the reading order is clear visually to sighted users as flowing from the top to the bottom of the first column, then to the top of the next column. But if the document is not properly tagged, a screen reader may read the document from top to bottom, across both columns, interpreting them as one column.

The simplest way to ensure correct reading order is to structure the document correctly in the authoring tool used to create the document, before conversion to tagged PDF. Note, however, that pages with complex layouts with graphics, tables, footnotes, side-bars, form fields, and other elements may not convert to tagged PDF in the correct reading order. These inconsistencies must then be corrected with repair tools such as Acrobat Pro.

When a PDF document containing form fields has a correct reading order, all form fields are contained in the tab order in the appropriate order, and in the correct order relative to other content in the PDF. Common tab-order errors include:

- Form fields missing from the tagged content.

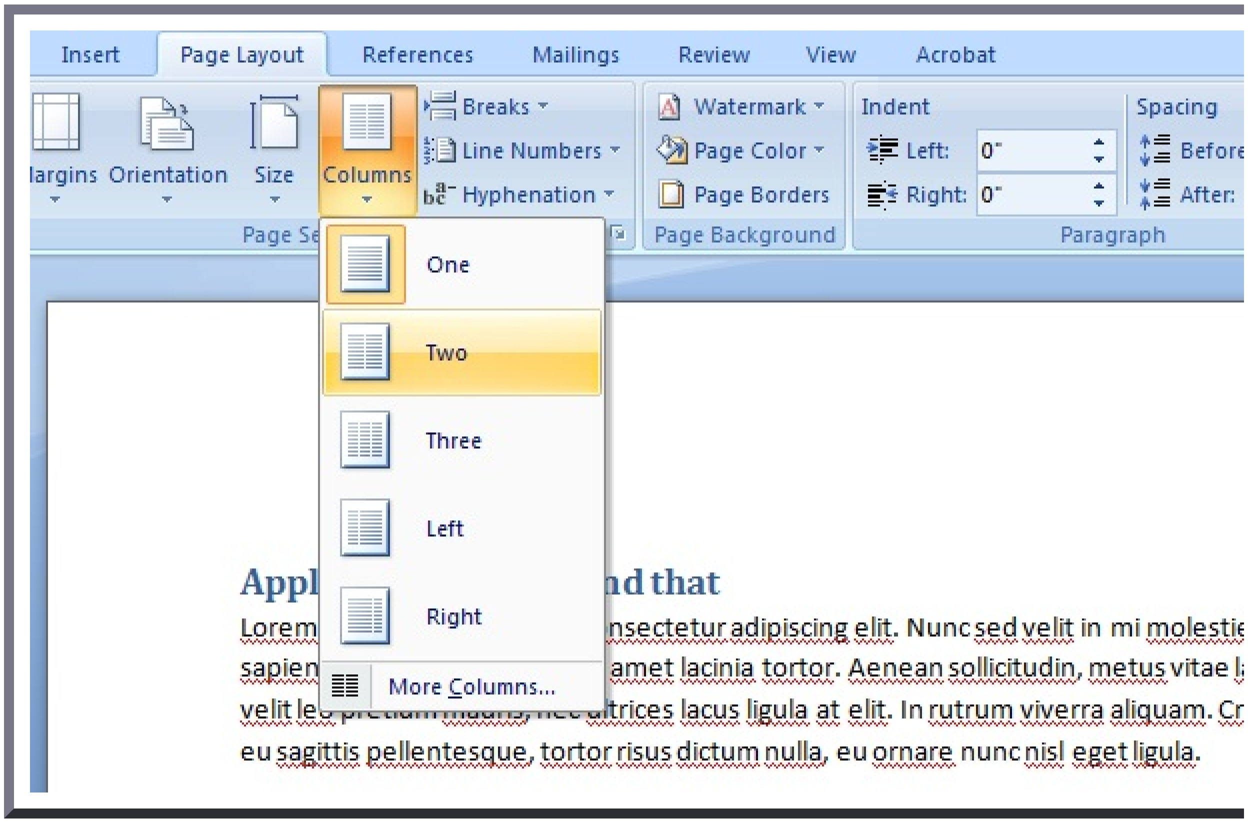
- Form fields in the wrong location in the PDF content; e.g., at the end of non-interactive content.

Examples

Example 1: Creating a 2-column document using Microsoft Word 2007

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in [PDF Authoring Tools that Provide Accessibility Support](#).

Multi-column documents created using Word's Page Layout > Columns... tool typically are in the correct reading order when converted to tagged PDF. The image below shows Word's Columns tool.



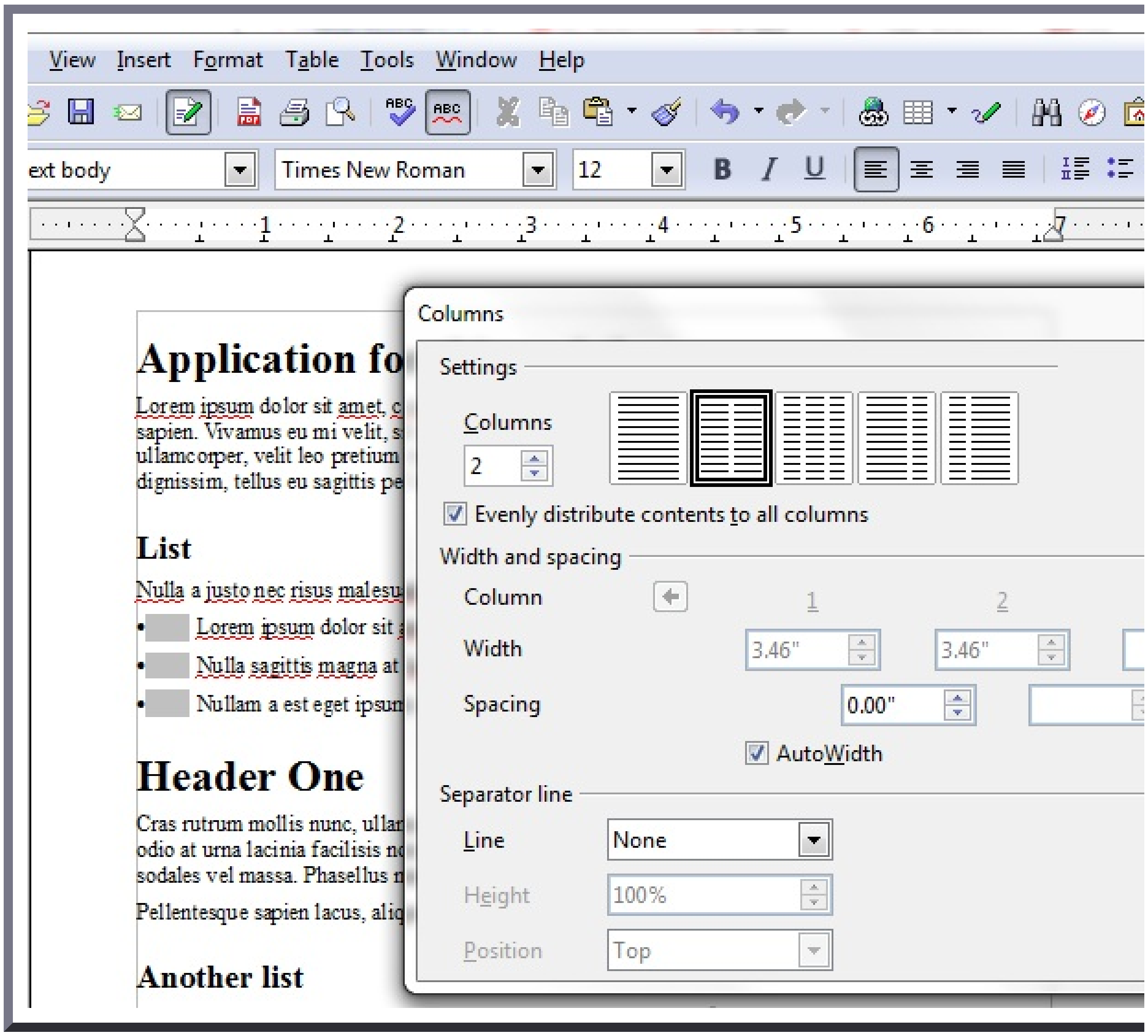
Lorem nsectetur adipiscing elit. Nunc sed velit in mi molestie
sapien amet lacinia tortor. Aenean sollicitudin, metus vitae l
velit le ltrices lacus ligula at elit. In rutrum viverra aliquam. Cr
eu sagittis pellentesque, tortor risus dictum nulla, eu ornare nunc nisl eget ligula.

This example is shown in operation in the [working example of 2-column document using Word 2007 (Word file)](#) and [working example of 2-column document using Word 2007 (PDF file)](#).

Example 2: Creating a 2-column document using OpenOffice.org Writer 2.2

This example is shown with OpenOffice.org Writer. There are other software tools that perform similar functions. See the list of other software tools in [PDF Authoring Tools that Provide Accessibility Support]().

Multi-column documents created using OpenOffice.org Writer's Format > Columns... tool typically are in the correct reading order when converted to tagged PDF. The image below shows Writer's Columns tool.



Columns: 2
[✓] Evenly distribute contents to all columns
Column 1 2
Width 3.46" 3.46"
Spacing 0.00"
[✓] AutoWidth
Line: None
Height: 100%
Position: Top

This example is shown in operation in the [working example of 2-column document using OpenOffice Writer (OpenOffice file)]() and [working example of 2-column document using OpenOffice Writer (PDF file)]().

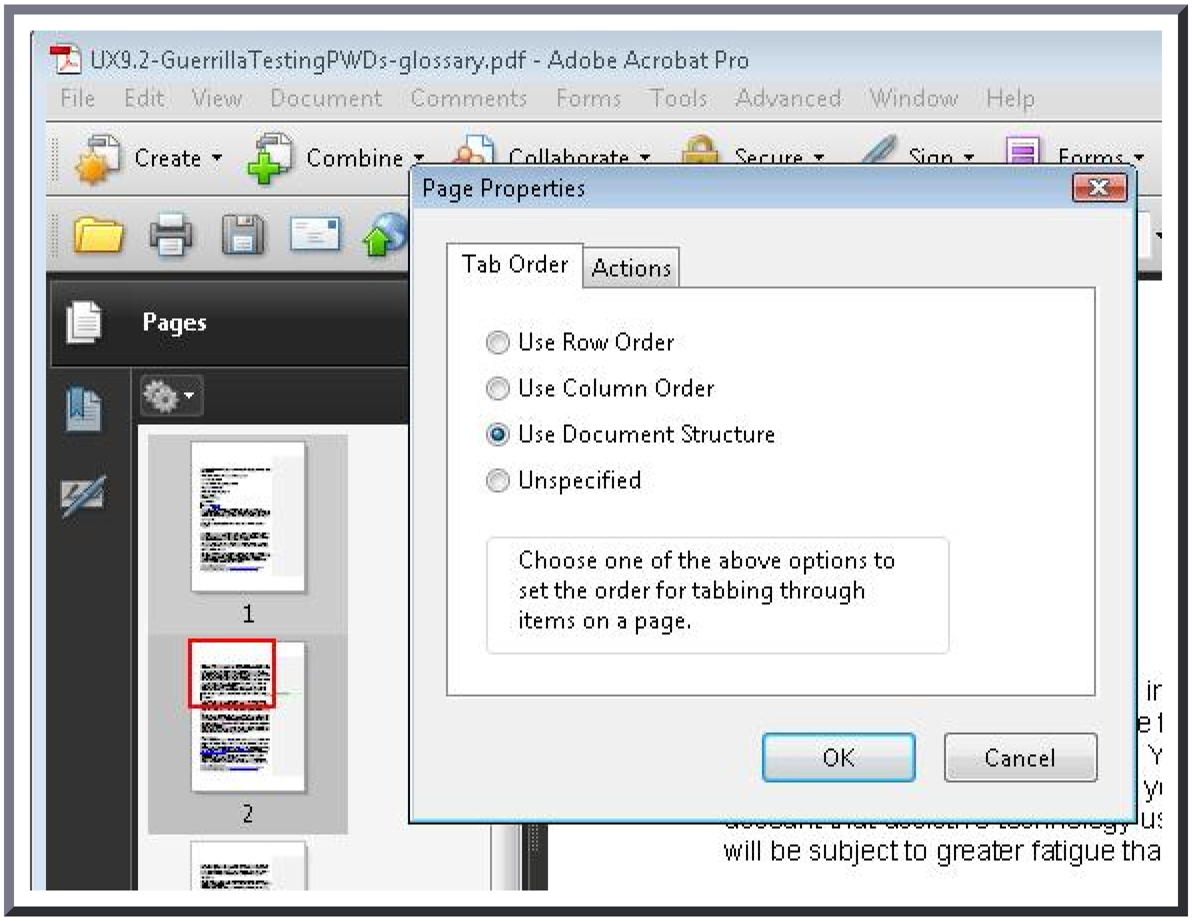
Example 3: Setting the tab order for one or more pages using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

In a tagged PDF document:

1. Open the Pages panel by either:
   - Clicking the Pages icon
   - Or selecting View > Navigation Panels > Pages
2. Select one or more page thumbnails.
3. Access the context menu for the selected thumbnail(s) and select Page Properties...
4. Select the Tab Order tab in the Page Properties dialog.
5. If needed, select a tab order option:

Option	**Description**
Use Row Order	Tabs from the upper left field, moving first left to right and then down, one table row at a time.
Use Column Order	Tabs from the upper left field, moving first from top to bottom and then across from left to right, one table column at a time.
Use Document Structure	For tagged documents, moves in the tag order specified by the authoring application. *Note:* This is usually the correct reading order and will be selected by default for tagged documents.
Unspecified	If the document was created using an earlier version of Acrobat Pro, the tab order is Unspecified by default. With this setting, form fields are tabbed through first, followed by links and then comments ordered by row. This may not be correct reading order.


UX9.2-GuerrillaTestingPWDs-glossary.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms
Pages
Page Properties
Tab Order
Actions
OK
Cancel

Use Row Order
Use Column Order
Use Document Structure [✓]
Unspecified

Choose one of the above options to set the order for tabbing through items on a page.

This example is shown in operation in the working example of setting the tab order (Word file) and working example of setting the tab order (PDF file).

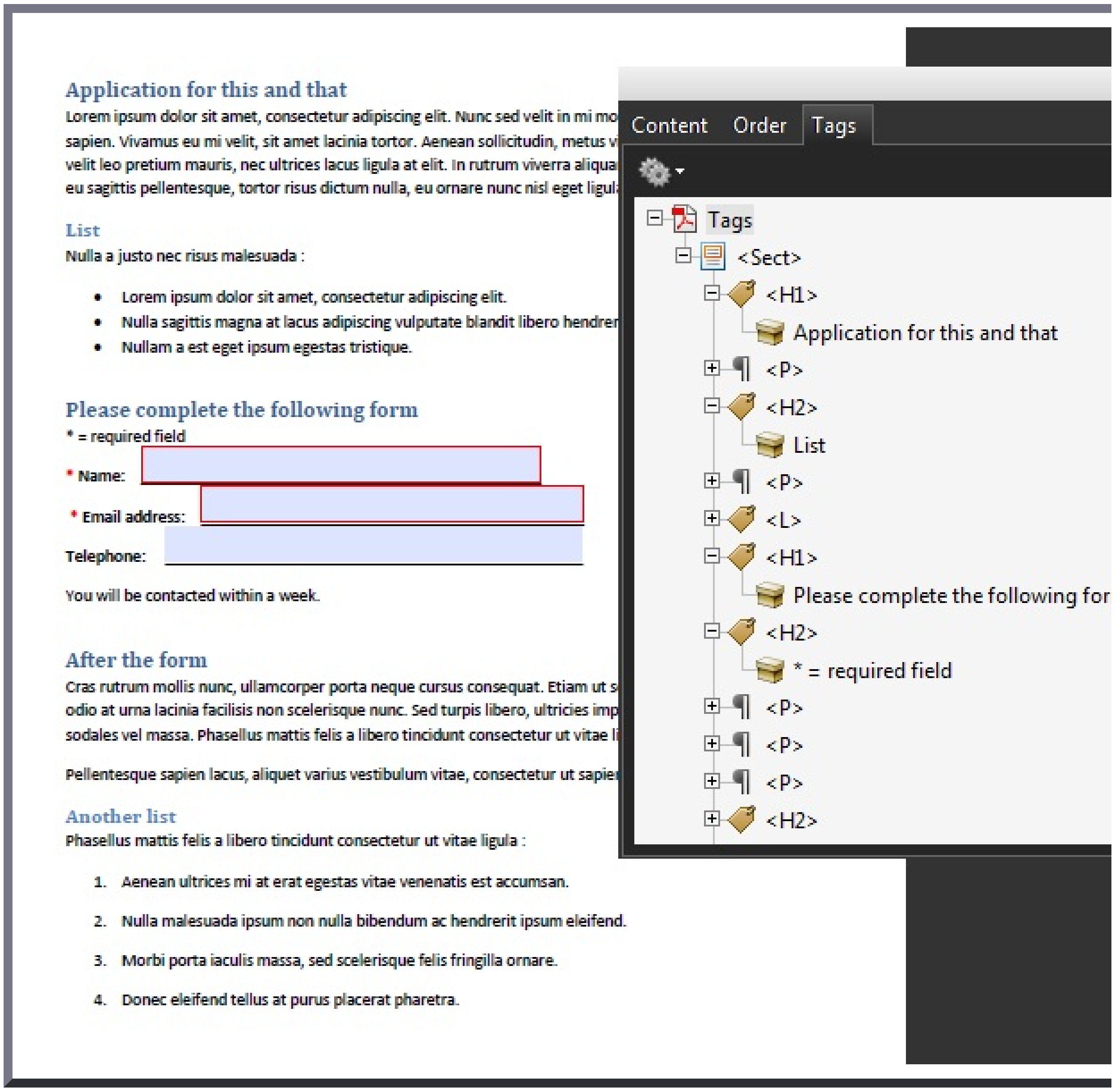
Example 4: Repairing the reading order using the Tags panel in Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support (http://trace.wisc.edu/wcag_wiki/index.php?title=PDF_Technology_Notes).

To correct the reading order in Example 5, use the Tags panel, and either

- Drag-and-drop the `H1` tag to precede the required-field text (tagged `H2`), or
- Cut-and-paste the `H2` tag to follow the `H1` tag.

In the following image, the reading order is correct for the text and header. That is, the content elements `H1` and `H2` have been switched into the correct reading order.



Application for this and that

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc sed velit in mi mo sapien. Vivamus eu mi velit, sit amet lacinia tortor. Aenean sollicitudin, metus v velit leo pretium mauris, nec ultrices lacus ligula at elit. In rutrum viverra aliqua eu sagittis pellentesque, tortor risus dictum nulla, eu ornare nunc nisl eget ligul

List

Nulla a justo nec risus malesuada :

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nulla sagittis magna at lacus adipiscing vulputate blandit libero hendrer
- Nullam a est eget ipsum egestas tristique.

Please complete the following form

* = required field

* Name:

* Email address:

Telephone:

You will be contacted within a week.

After the form

Cras rutrum mollis nunc, ullamcorper porta neque cursus consequat. Etiam ut s odio at urna lacinia facilisis non scelerisque nunc. Sed turpis libero, ultricies imp sodales vel massa. Phasellus mattis felis a libero tincidunt consectetur ut vitae l

Pellentesque sapien lacus, aliquet varius vestibulum vitae, consectetur ut sapie

Another list

Phasellus mattis felis a libero tincidunt consectetur ut vitae ligula :

1. Aenean ultrices mi at erat egestas vitae venenatis est accumsan.
2. Nulla malesuada ipsum non nulla bibendum ac hendrerit ipsum eleifend.
3. Morbi porta iaculis massa, sed scelerisque felis fringilla ornare.
4. Donec eleifend tellus at purus placerat pharetra.

```
Tags
  <Sect>
    <H1>
      Application for this and that
    <P>
    <H2>
      List
    <P>
    <L>
    <H1>
      Please complete the following for
    <H2>
      * = required field
    <P>
    <P>
    <P>
    <H2>
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8 (Tagged PDF) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility
- Making PDF documents accessible with Adobe Acrobat Pro

Related Techniques

- G57: Ordering the content in a meaningful sequence
- G59: Placing the interactive elements in an order that follows sequences and relationships within the content
- G202: Ensuring keyboard control for all functionality

Tests

Procedure

1. Verify that the content is in the correct reading order by one of the following:
    - Read the PDF document with a screen reader or a tool that reads aloud, listening to hear that each element is read in the correct order.
    - Use a tool that exposes the document through the accessibility API, and verify that the reading order is correct.
2. Verify that the tab order is correct for focusable content by one of the following:
    - Use the tab key to traverse the focus order in the document.
    - Use a tool that is capable of showing the page object entry that specifies the tab order setting to open the PDF document and view the setting.

Expected Results

- #1 and Check #2 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF4: Hiding decorative images with the Artifact tag in PDF documents

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 1.1.1 (Non-text Content)
  - How to Meet 1.1.1 (Non-text Content)
  - Understanding Success Criterion 1.1.1 (Non-text Content)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF4. Also see PDF Technology Notes.

Description

The purpose of this technique is to show how purely decorative images in PDF documents can be marked so that they can be ignored by Assistive Technology by using the /Artifact tag. This is typically accomplished by using a tool for authoring PDF.

In PDF, artifacts are generally graphics objects or other markings that are not part of the authored content. Examples of artifacts include page header or footer information, lines or other graphics separating sections of the page, or decorative images.

Examples

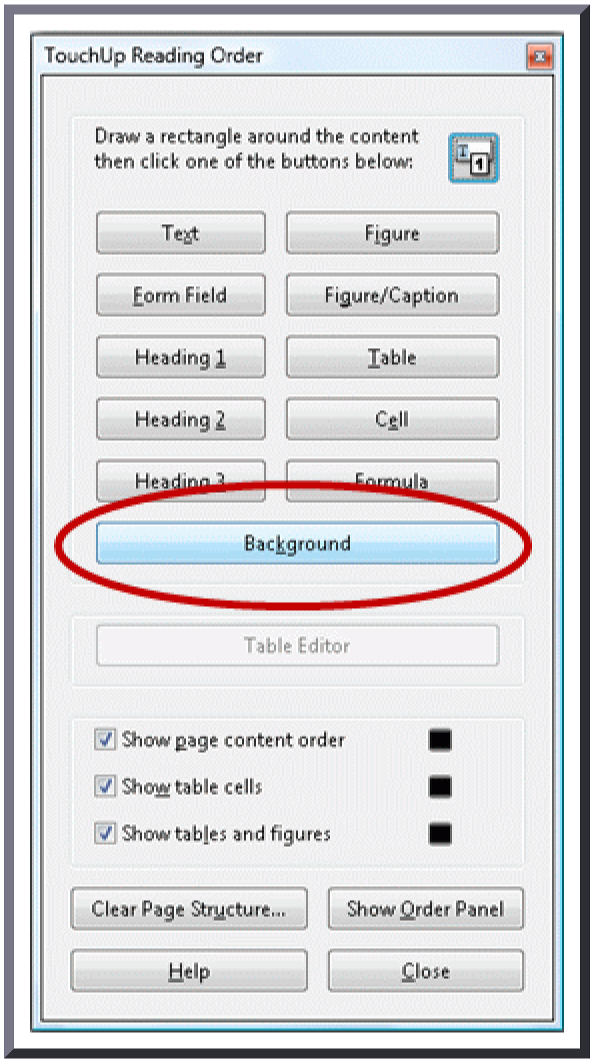
Example 1: Marking a background image as an artifact using Adobe Acrobat 9 Pro's TouchUp Reading Order Tool

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

The TouchUp Reading Order Tool can be used to mark an image as "Background," which removes it from the document tag structure.

1. Open the TouchUp Reading Order Tool in Acrobat Pro: Advanced > Accessibility > TouchUp Reading Order
2. Select the decorative image in the document
3. In the TouchUp Reading Order Tool, click the Background button to remove the selected image from the tag structure

The screenshot below illustrates this example.



Draw a rectangle around the content then click one of the buttons below:

[✓] Show page content order
[✓] Show table cells
[✓] Show tables and figures

This example is shown in operation in the working example of creating a decorative image (Word file) and working example of marking a background image as an artifact (PDF file).

Example 2: Marking an image as an artifact in a PDF document using an /Artifact tag or property list

The PDF specification allows images to be marked as "artifacts" as defined in Section 14.8.2.2 (Real Content and Artifacts) in PDF 1.7 (ISO 32000-1). An artifact is explicitly distinguished from real content by enclosing it in a marked-content sequence with the /Artifact tag.

/Artifact

```
BMC  ...  EMC
```

or

/Artifact propertyList

```
BDC  ...  EMC
```

The first is used to identify a generic artifact; the second is used for artifacts that have an associated property list. Note, to aid in text reflow, artifacts should be defined with property lists whenever possible. Artifacts lacking a specified bounding box are likely to be discarded during reflow.

Property list entries for artifacts include Type, BBox, Attached, and Subtype.

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8.2.2 (Real Content and Artifacts) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Tests

Procedure

1. For an image that is purely decorative, use one of the following to verify that it is marked as an artifact:

- Read the PDF document with a screen reader, listening to hear that the decorative image is not announced when reading the content line-by-line.
- Using a PDF editor, make sure the decorative image is marked as an artifact.
- Reflow the document and make sure the decorative image does not appear on the page.
- Use a tool that is capable of showing the /Artifact entry or property list, such as aDesigner, to open the PDF document and verify that decorative images are marked as artifacts.
- Use a tool that exposes the document through the accessibility API and verify that the decorative image is not exposed through the API.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF5: Indicating required form controls in PDF forms

Applicability

Tagged PDF documents with forms

This technique relates to:

- Success Criterion 3.3.1 (Error Identification)
  - How to Meet 3.3.1 (Error Identification)
  - Understanding Success Criterion 3.3.1 (Error Identification)
- Success Criterion 3.3.2 (Labels or Instructions)
  - How to Meet 3.3.2 (Labels or Instructions)
  - Understanding Success Criterion 3.3.2 (Labels or Instructions)

  Note: This technique must be combined with other techniques to meet SC 3.3.2. See Understanding SC 3.3.2 for details.

- Success Criterion 3.3.3 (Error Suggestion)
  - How to Meet 3.3.3 (Error Suggestion)
  - Understanding Success Criterion 3.3.3 (Error Suggestion)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF5. Also see PDF Technology Notes.

Description

The objective of this technique is to notify the user when a field that must be completed has not been completed in a PDF form. Required fields are implemented using the /Ff entry in the form field's dictionary (see Table 220 in Section 12.7 (Interactive Forms) of PDF 1.7 (ISO 32000-1). This is normally accomplished using a tool for authoring PDF.

If errors are found, an alert dialog describes the nature of the error in text. This may be accomplished through scripting created by the author (see, for example, SCR18: Providing client-side validation and alert). User agents, such as Adobe Acrobat Pro and LiveCycle, can provide automatic alerts (as described in the examples below).

Note: once the user dismisses the alert dialog, it may be helpful if the script positions the keyboard focus on the field where the error occurred, although some users may expect the focus to remain on the last control focused prior to the alert appearing. Authors should exercise care to ensure that any movement of the focus will be expected. For example, if the alert announces a missing required phone number, positioning the focus on the phone number field when the alert is dismissed can be regarded as helpful and expected. In some cases, however, this may not be possible. If multiple input errors occur on the page, another approach must be taken to error reporting. (See, for example, the Adobe scripting resources.)

Ensuring that users are aware an error has occurred, can determine what is wrong, and can correct it are keys to software usability and accessibility. Meeting this objective helps ensure that all users can complete transactions with ease and confidence.

Labels for required form controls

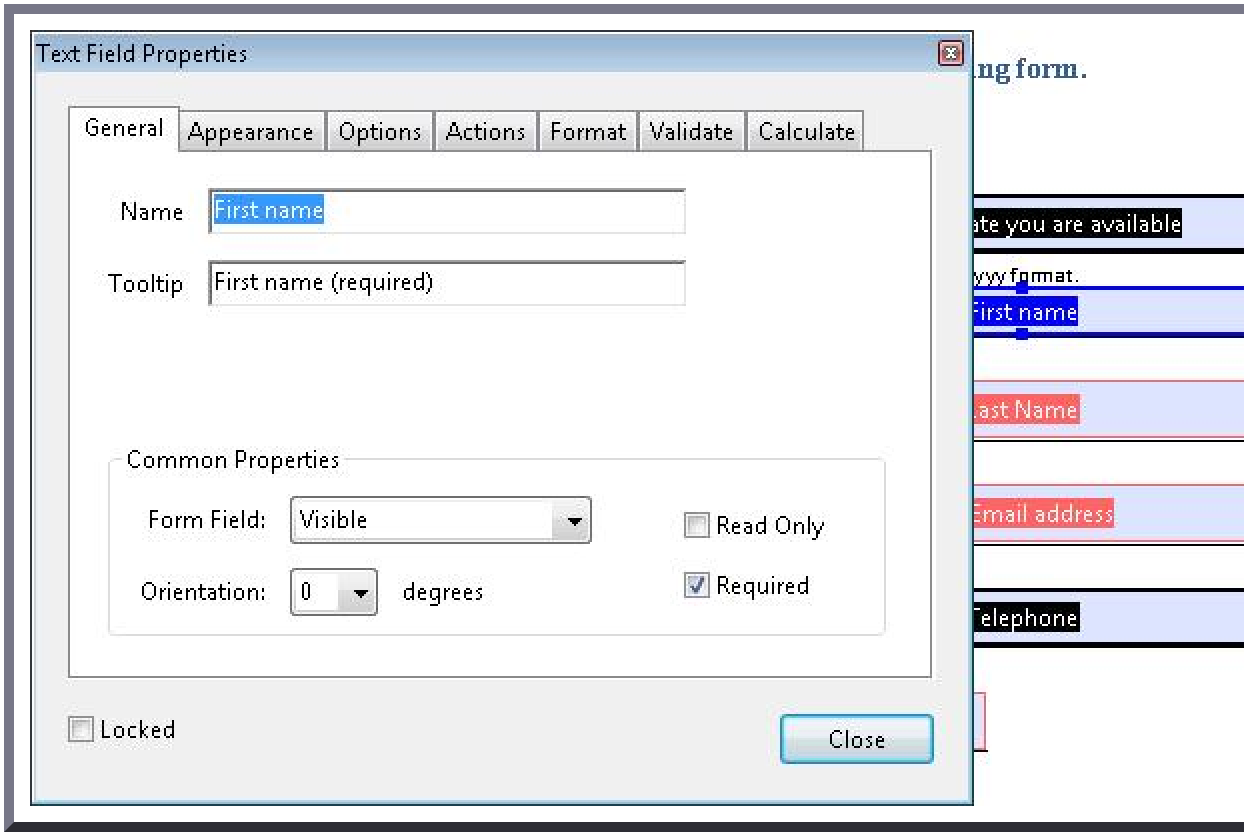
It is also important that users are aware that an error *may* occur. You can incorporate this information in labels; for example, "Date (required)" or the use of a red asterisk to indicate required fields. (Make sure that a legend appears on each form with required fields, e.g., "* = required field".) See *PDF10: Providing labels for interactive form controls in PDF documents*.

Examples

Example 1: Creating a required field in a PDF form using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Access the context menu of the field and select the Properties dialog.
2. If the field is required, check the Required box. This checkbox forces the user to fill in the selected form field. If the user attempts to submit the form while a required field is blank, an error message appears and the empty required form field is highlighted.



Text Field Properties

General | Appearance | Options | Actions | Format | Validate | Calculate

Name: First name

Tooltip: First name (required)

Common Properties

Form Field: Visible

Orientation: 0 degrees

Read Only: unchecked

Required: ✓

Locked: unchecked

This example is shown in operation in the working example of creating a required field in Acrobat.

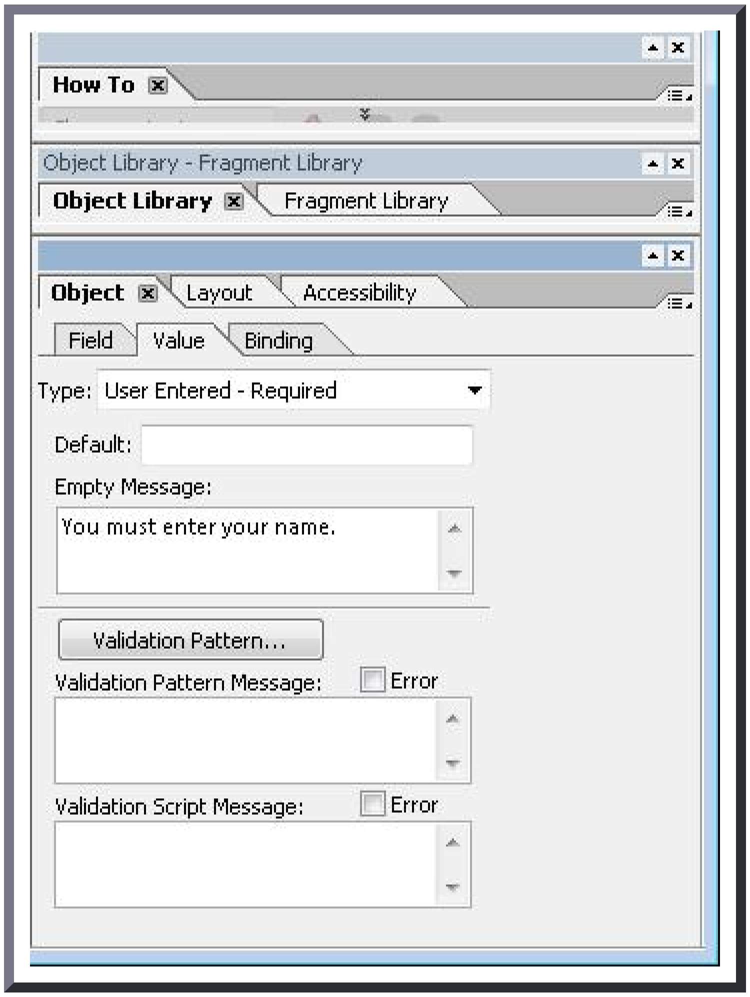
Example 2: Creating a required field in a PDF form using Adobe LiveCycle Designer ES 8.2.1

This example is shown with Adobe LiveCycle Designer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Access the context menu of the form control, select Palettes, and select Object.
2. Select "User entered - Required" from the Type pulldown.
3. Enter an error message in the "Empty Message" field. This message appears when a user tries to submit the form without entering a value in the required field.

If the user attempts to submit the form with a required field left blank, the Empty Message text appears and the empty required field is highlighted.

The image below shows the Adobe LiveCycle Object palette with the required selection.



Type: User Entered - Required

Default:

Empty Message:

You must enter your name.

Validation Pattern Message:

Validation Script Message:

You can also add explicit text to the form label to indicate required fields (e.g., "(Required)").

This example is shown in operation in the working example of creating a required field in LiveCycle Designer.

Example 3: Adding a required text field in a PDF form using the /Tx field type and Ff flag

The following code fragment illustrates code that is typical for the object definitions for a typical text field. Note that the text field is required, using the Ff flag. This is typically accomplished by an authoring tool.

```
<< /AP -dict-
   /DA /Helv  0 Tf 0 g
   /DR -dict-
   /F 0x4
   /FT Tx              % FT key set to Tx for Text Field
   /Ff 0x2             % Ff integer 0x2 value indicates required
   /P -dict-
   /Rect -array-
   /StructParent 0x1
   /Subtype Widget
   /T First            % Partial field name First
   /TU First name (required)  % TU tool tip value serves as short description
   /Type Annot
   /V Pat Jones
>>
...
<Start Stream>
 BT
  /P <</MCID 0 >>BDC
  /CS0 cs 0  scn
  /TT0 1 Tf
    -0.001 Tc 0.003 Tw 11.04 0 0 11.04 72 709.56 Tm
    [(P)-6(le)-3(as)10(e)-3( )11(P)-6(rin)2(t)-3( Y)8(o)-7(u)2(r N)4(a)11(m)-6(e)]TJ
  0 Tc 0 Tw 9.533 0 Td
  ( )Tj
  -0.004 Tc 0.004 Tw 0.217 0 Td
  [(\()-5(R)-4(e)5(q)-1(u)-1(i)-3(r)-3(e)-6(d)-1(\))]TJ
 EMC
  /P <</MCID 1 >>BDC
  0 Tc 0 Tw 4.283 0 Td
  [( )-2( )]TJ
   EMC
   /ArtifactSpan <</MCID 2 >>BDC
   0.002 Tc -0.002 Tw 0.456 0 Td
  [(__)11(___)11(___)11(___)11(___)11(_)11(____)11(___)11(___)11(__)]TJ
  0 Tc 0 Tw 13.391 0 Td
  ( )Tj
  EMC
 ET
<End Stream>
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 12.7 (Interactive Forms) in PDF 1.7 (ISO 32000-1)
- Adobe XML Forms Architecture (XFA)
- PDF and Accessibility

Related Techniques

- G83: Providing text descriptions to identify required fields that were not completed
- H90: Indicating required form controls using label or legend
- SCR18: Providing client-side validation and alert
- PDF23: Providing interactive form controls in PDF documents
- PDF10: Providing labels for interactive form controls in PDF documents
- PDF22: Indicating when user input falls outside the required format or values in PDF forms

Tests

Procedure

For each form field that is required, verify that validation information and instructions are provided by applying the following:

1. Check that the required status is indicated in the form control's label.
2. Leave the field blank and submit the form. Check that an alert describing the error is provided.
3. Use a tool that exposes the document through the accessibility API, and verify that the required property is indicated.

Expected Results

- #1, #2, and #3 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF6: Using table elements for table markup in PDF Documents

Applicability

Tagged PDF documents with tables

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
  - How to Meet 1.3.1 (Info and Relationships)
  - Understanding Success Criterion 1.3.1 (Info and Relationships)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF6. Also see PDF Technology Notes.

Description

The purpose of this technique is to show how tables in PDF documents can be marked up so that they are recognized by assistive technology. This is typically accomplished by using a tool for authoring PDF.

Tabular information must be presented in a way that preserves relationships within the information even when users cannot see the table or the presentation format is changed. Information is considered tabular when logical relationships among text, numbers, images, or other data exist in two dimensions (vertical and horizontal). These relationships are represented in columns and rows, and the columns and rows must be recognizable in order for the logical relationships to be perceived.

Tagged tables can be created using the Add Tags to Document feature in Adobe Acrobat, using the Object Library in Adobe LiveCycle, or converting tables to PDF from a third-party application, such as Microsoft Word. However, the resulting tables may not be tagged correctly and you should ensure that table tagging issues are resolved.

Within PDF documents, a table uses the following structure types for table elements:

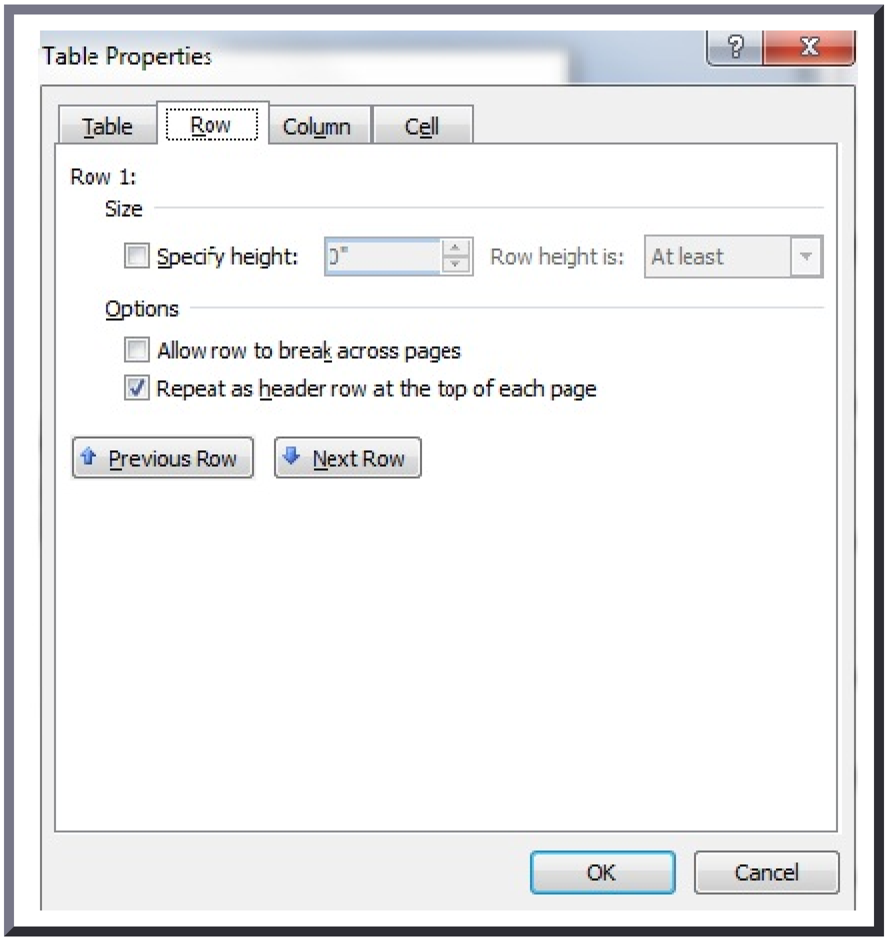
- A table element (*Table*).
- One or more table row elements(*TR*) which define each row of table cells as immediate children of the *Table* element.
- One or more table header elements (*TH*) or table data elements (*TD*) as the immediate children of each table row element.
- Cells that span two or more rows or columns should use the *RowSpan* or *ColSpan* attribute.
- For tables that contain blank cells, you may need to add empty *TD* cells so that each row or column has the same number of cells.

Examples

Example 1: Creating tables in Microsoft Word 2007 that have correctly tagged headings when converted to PDF

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Access the table header row's context menu and select Table Properties...
2. Select the Row tab.
3. Check "Repeat as header at the top of each page" as shown in the following image.



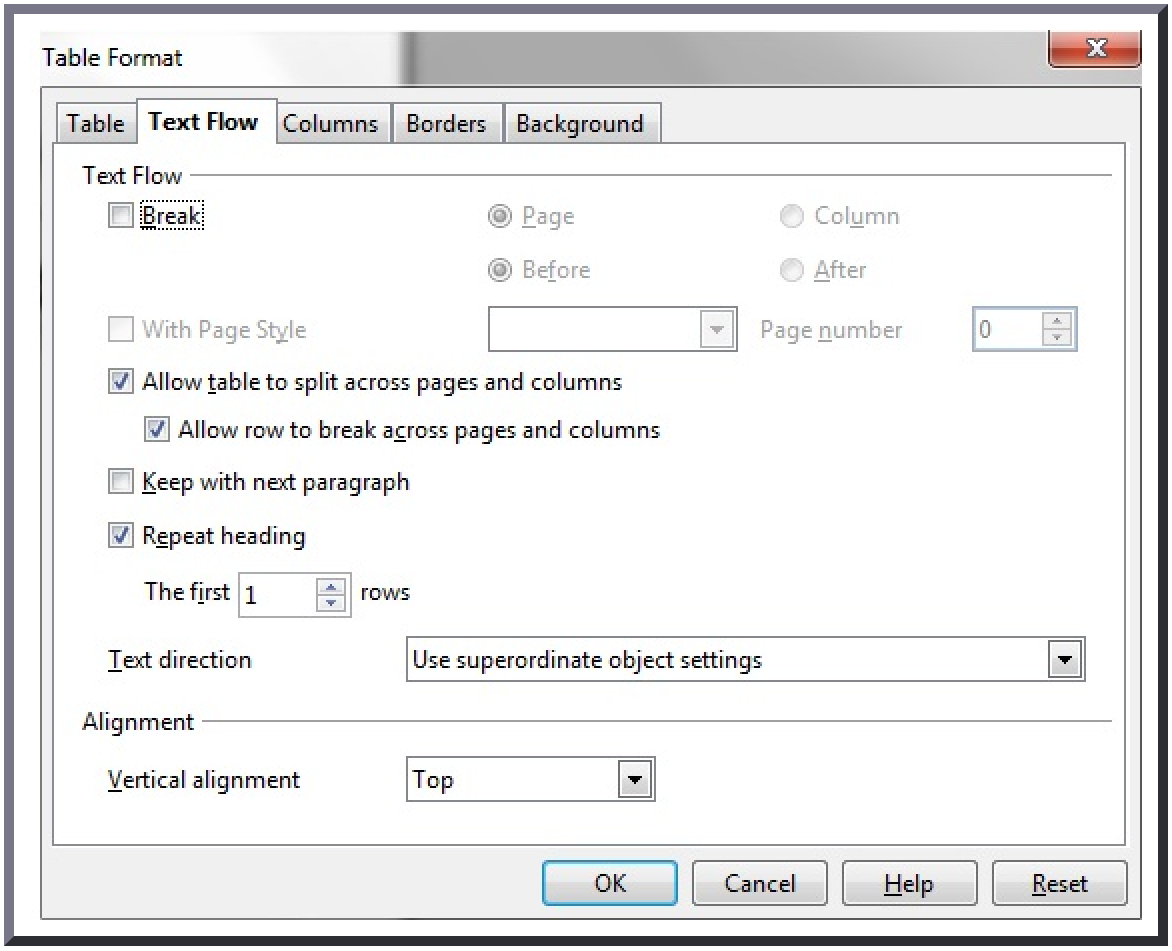
This example is shown in operation in the working example of tagged table headings in Word 2007.

Note: Microsoft Word can only mark up cells as column headings, not as row headings. Only the first row can be marked as heading for all table columns. When the table has row headings or a more complex heading structure, this mark-up must be added in a PDF editor such as Acrobat Pro.

Example 2: Creating tables in OpenOffice.org Writer 2.2 that have correctly tagged headings when converted to PDF

This example is shown with OpenOffice.org Writer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Access the table's context menu and select Table...
2. Select the Table Format tab.
3. Check Repeat Heading and select "1" in the First Rows listbox as shown in the following image.



Text Flow

[ ] Break ● Page ○ Column
● Before ○ After

[ ] With Page Style Page number 0

[✓] Allow table to split across pages and columns

[✓] Allow row to break across pages and columns

[ ] Keep with next paragraph

[✓] Repeat heading

The first 1 rows

Text direction: Use superordinate object settings

Alignment

Vertical alignment: Top

This example is shown in operation in the working example of tagged table headings in OpenOffice Writer.

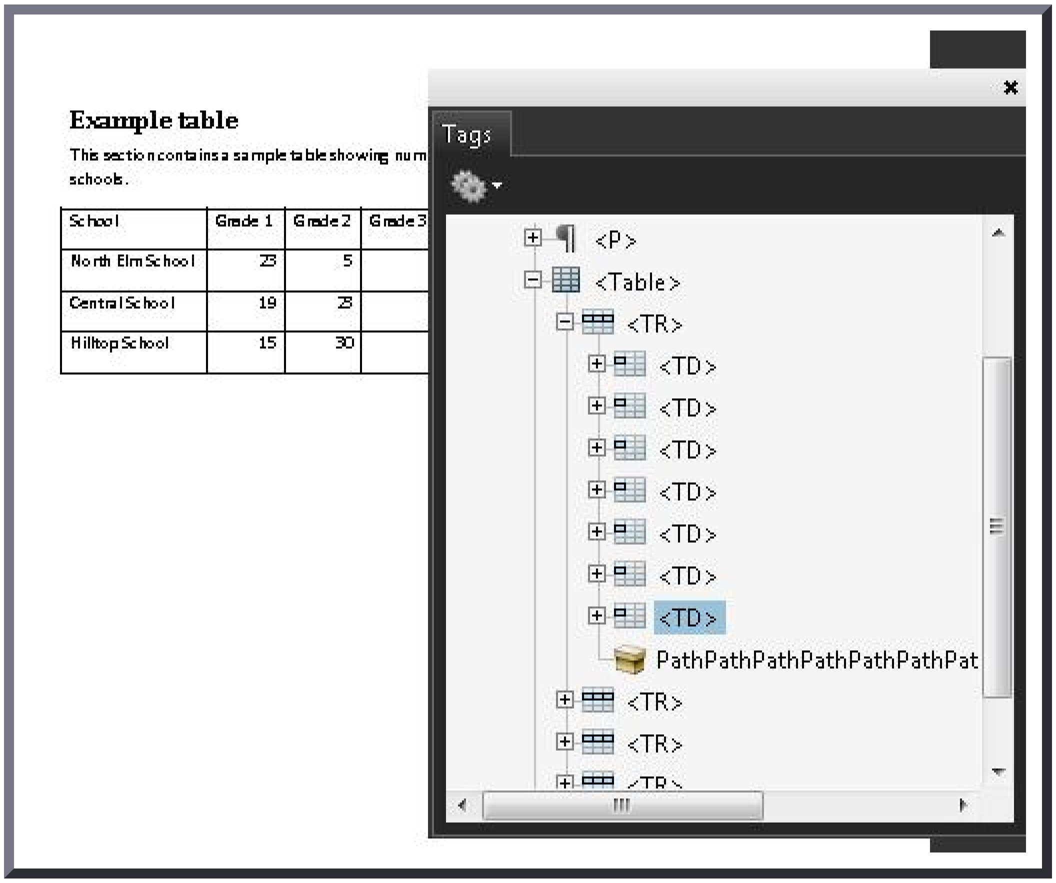
Note: OpenOffice.org Writer can only mark up cells as column headings, not as row headings. Only the first row can be marked as heading for all table columns. When the table has row headings or a more complex heading structure, this mark-up must be added in a PDF editor such as Acrobat Pro.

Example 3: Modifying table tags using the Tags tab in Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

To check that a converted document with tables has correct table tagging:

- In the View menu, select Navigation Panel, then select Tags.



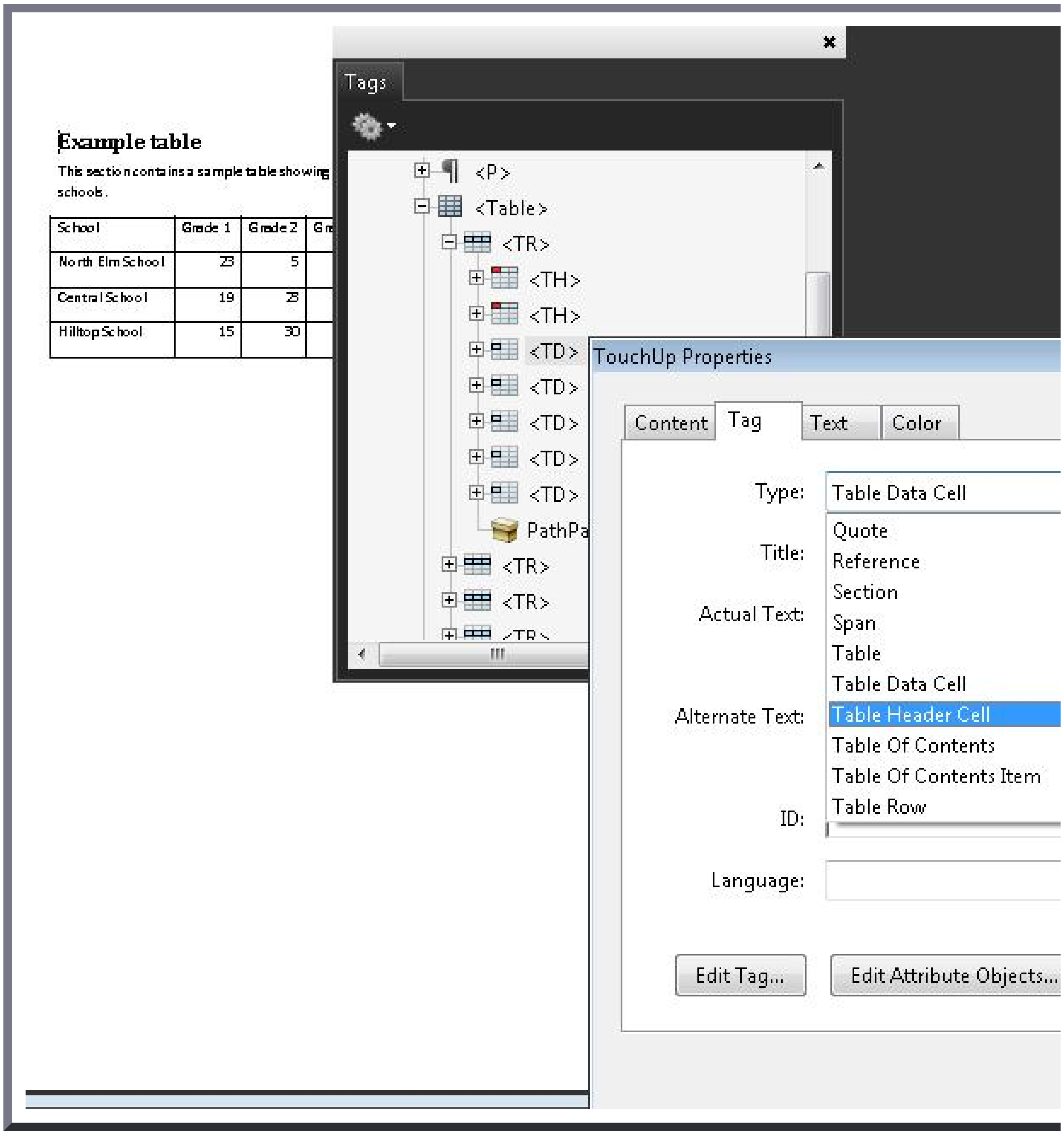
Example table

This section contains a sample table showing num
schools.

School	Grade 1	Grade 2	Grade 3
North Elm School	23	5	
Central School	19	23	
Hilltop School	15	30	

Note that in this case, the table headers were not formatted as illustrated in Examples 1 and 2, and are marked as data cells (*TD*). To change these to *TH* tags:

1. On the Tags tab, open the table row that contains the header cells, as shown on the image above.
2. Select on the first data cell and select Properties...
3. On the Tags tab in the Properties dialog, use the Type dropdown to change Table Data Cell to Table Header Cell.
4. Repeat for all the table header cells in the first table row.



Example table

This section contains a sample table showing
schools.

School	Grade 1	Grade 2	Gr
North Elm School	23	5	
Central School	19	23	
Hilltop School	15	30	

Tags

```
<P>
<Table>
  <TR>
    <TH>
    <TH>
    <TD>
    <TD>
    <TD>
    <TD>
    <TD>
    PathPa
  <TR>
  <TR>
  <TR>
```

TouchUp Properties

Content | Tag | Text | Color

Type: Table Data Cell

Title:

Actual Text:

Alternate Text:

ID:

Language:

Quote
Reference
Section
Span
Table
Table Data Cell
Table Header Cell
Table Of Contents
Table Of Contents Item
Table Row

Edit Tag... | Edit Attribute Objects...

This example is shown in operation in the working example of tagged table headings in Acrobat.

Example 4: Marking up a table using table structure elements

The following code fragment illustrates code that is typical for a simple table (header row and data row) such as shown in Examples 1-3:

```
95 0 obj                %Structure element for a table
 <<
  /A 39 0 R
  /K[96 0 R 101 0 R 106 0 R 111 0 R]
  /P 93 0 R
  /S/Table              %standard structure type is table
 >>
 endobj
96 0 obj                %Structure element for a table row
 <<
  /K[97 0 R 98 0 R 99 0 R 100 0 R]
  /P 95 0 R
  /S/TR                 %standard structure type is table row
 >>
 endobj
97 0 obj                %Structure element for a table header
 <</A[23 0 R 120 0 R]
   /K 1
   /P 96 0 R
   /S/TH                 %standard structure type is table head
   /Pg 8 0 R
 >>
endobj
104 0 obj                %Structure element for table data (cell contents)
 <<
  /A 29 0 R
  /K 7
  /P 101 0 R
  /S/TD                  %standard structure type is table data
  /Pg 8 0 R
 >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8.4.3.4 (Table Elements) in PDF 1.7 (ISO 32000-1)

- PDF and Accessibility

Related Techniques

- H51: Using table markup to present tabular information
- PDF20: Using Adobe Acrobat Pro's Table Editor to repair mistagged tables

Tests

Procedure

1. For each table, confirm one of the following:
    - Read the PDF document with a screen reader, listening to hear that the tabular information is presented in a way that preserves logical relationships among the table header and data cells.
    - Using a PDF editor, verify that the appropriate *TR*, *TH*, and *TD* tags are in the proper reading order and hierarchy in the table tree.
    - Use a tool which is capable of showing the table elements to open the PDF document, view the table structure, and verify that it contains the appropriate TR, TH, and TD structures.
    - Use a tool that exposes the document through the accessibility API, and verify that the table structure contains the appropriate TR, TH, and TD structures, and that they are in the proper reading order and hierarchy.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF7: Performing OCR on a scanned PDF document to provide actual text

Applicability

Scanned PDF documents

This technique relates to:

- Success Criterion 1.4.5 (Images of Text)
    - How to Meet 1.4.5 (Images of Text)
    - Understanding Success Criterion 1.4.5 (Images of Text)
- Success Criterion 1.4.9 (Images of Text (No Exception))
    - How to Meet 1.4.9 (Images of Text (No Exception))
    - Understanding Success Criterion 1.4.9 (Images of Text (No Exception))

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF7. Also see PDF Technology Notes.

Description

The intent of this technique is to ensure that visually rendered text is presented in such a manner that it can be perceived without its visual presentation interfering with its readability.

A document that consists of scanned images of text is inherently inaccessible because the content of the document is images, not searchable text. Assistive technologies cannot read or extract the words; users cannot select, edit, resize, or reflow text nor can they change text and background colors; and authors cannot manipulate the PDF for accessibility.

For these reasons, authors should use actual text rather than images of text, using an authoring tool such as Microsoft Word or Oracle Open Office to author and convert content to PDF.

If authors do not have access to the source file and authoring tool, scanned images of text can be converted to PDF using optical character recognition (OCR). Adobe Acrobat Pro can then be used to create accessible text.

Examples

Example 1: Generating actual text rather than images of text using Adobe Acrobat 9 Pro

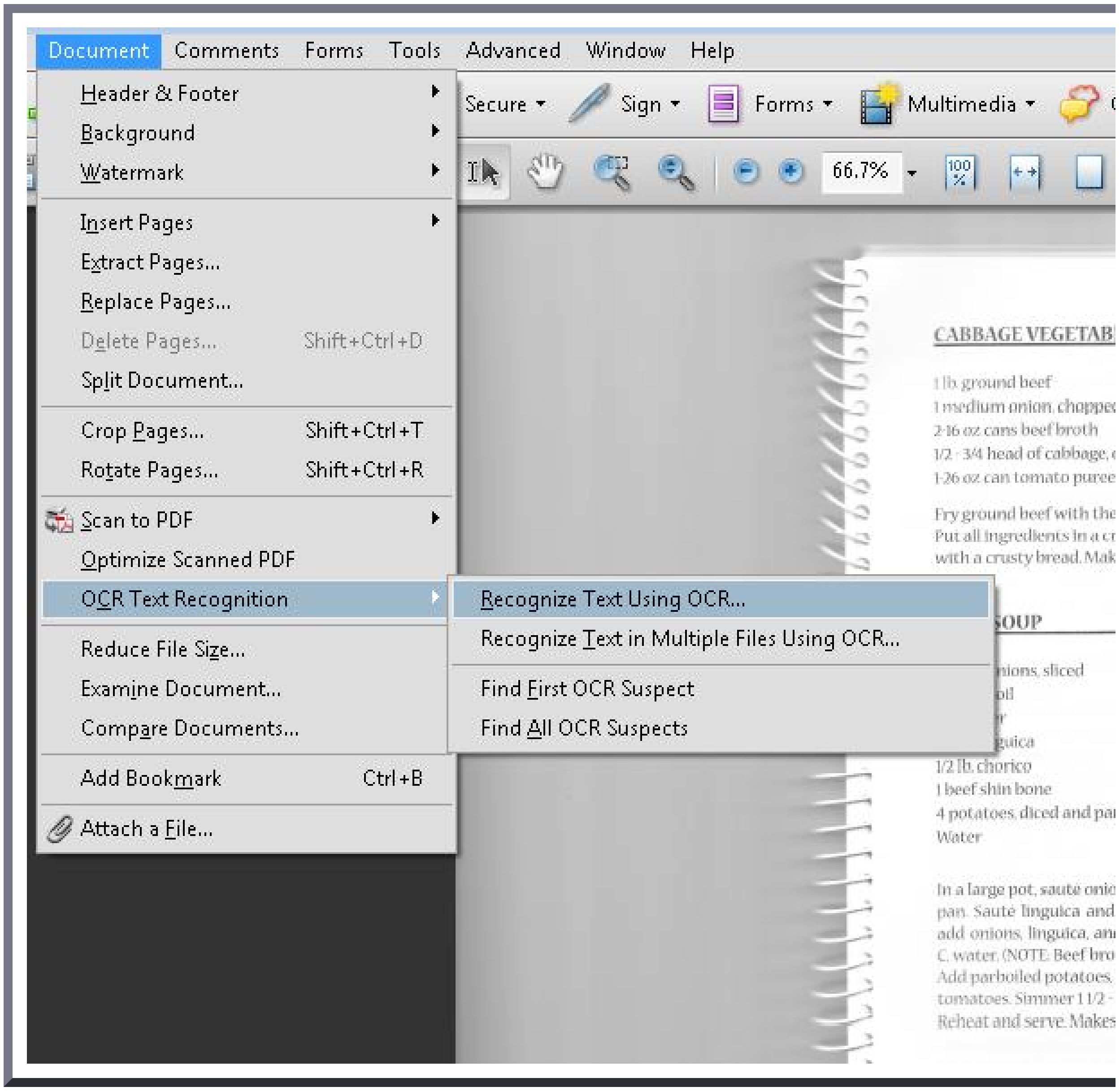
This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

This example uses a simple one-page scanned image of text. To ensure that actual text is stored in the document, perform the following steps:

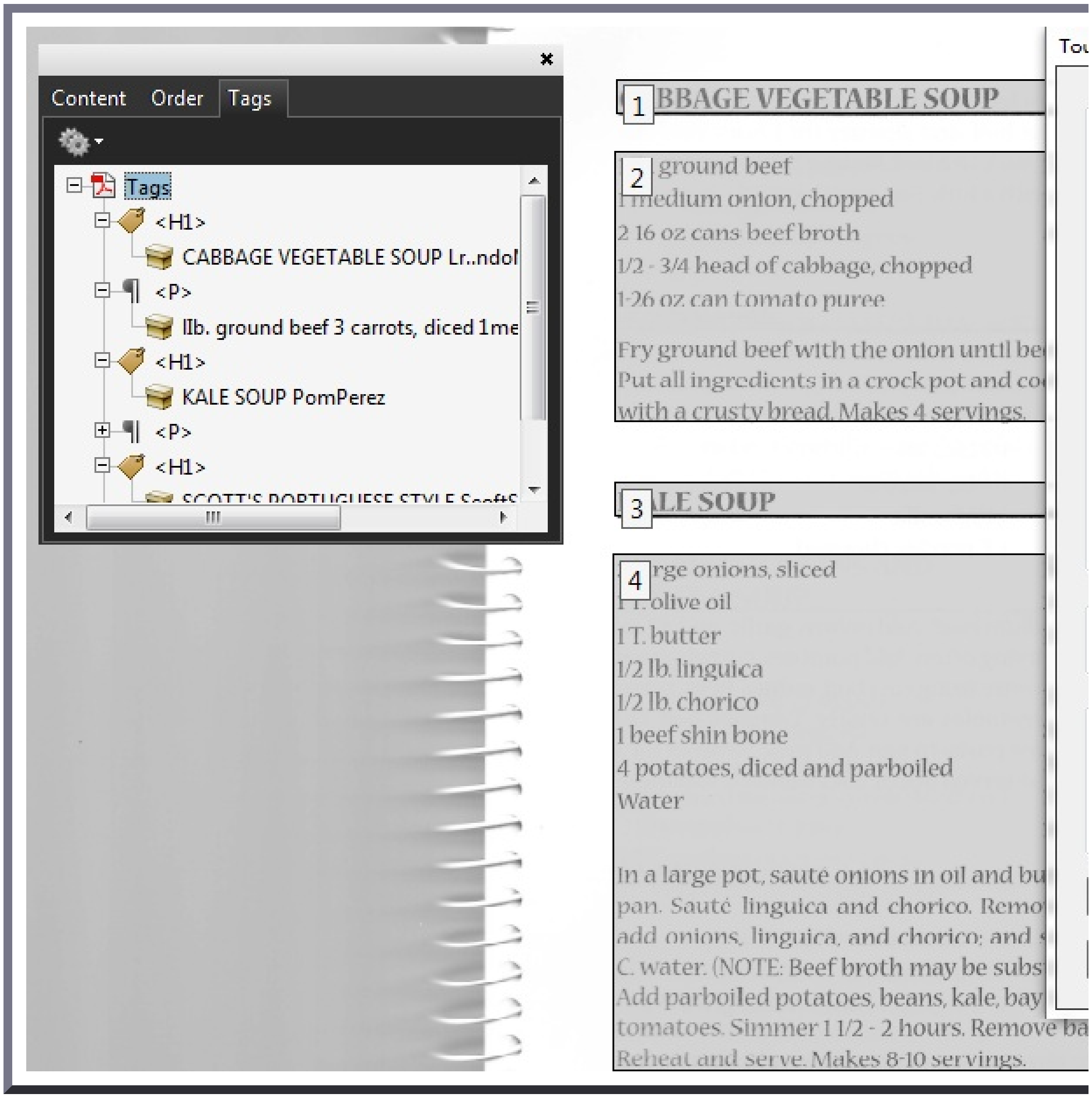
1. Scan the document using as high a resolution as possible to improve the OCR performance.
2. Load the scanned document in Acrobat Acrobat Pro. Select Document > OCR Text Recognition > Recognize Text Using OCR...
3. In the next dialog, select the All Pages radio button under Pages (or Current Page if you are converting only one page), and then select OK.
4. Under the Settings list, select Edit. In the next dialog, select Formatted Text and Graphics in the PDF Output Style drop-down list. This is important for ensuring accessibility.
5. Depending on the resolution and how clear the text was, OCR converts images of words and characters to actual text. Text that Acrobat Pro does not recognize is listed as an "OCR suspect," or text element that Acrobat suspects was not recognized correctly.
6. To fix the suspects, choose Document > OCR Text Recognition > Find First OCR Suspect. Acrobat Pro presents each suspect one at a time, which can be corrected using Acrobat Pro touchup tools.
7. Run Advanced > Accessibility > Add Tags to Document
8. Test for accessibility: Advanced > Accessibility > Full Check...

Note: Alternatively, you can use Document > OCR Text Recognition > Find All OCR Suspects to display all OCR suspects at the same time for faster editing.

The following image shows a scanned one-page document in Adobe Acrobat Pro.



The next image shows the converted content after adding tags to the document. It will probably be necessary to use the TouchUp Reading Order tool and the Tags panel to tag the content properly for the intended final document. For this example, the image of the spiral book binding was tagged in the conversion. The TouchUp Reading Order tool was used to hide the image as a background (decorative) image (see *PDF4: Hiding decorative images with the Artifact tag in PDF documents*). The recipe titles were tagged as first level headers.


Content
Order
Tags
Tags
<H1>
CABBAGE VEGETABLE SOUP Lr..ndol
<P>
llb. ground beef 3 carrots, diced 1me
<H1>
KALE SOUP PomPerez
<P>
<H1>
SCOTT'S PORTUGUESE STYLE ScottS
1
BBAGE VEGETABLE SOUP
2
ground beef
1 medium onion, chopped
2 16 oz cans beef broth
1/2 - 3/4 head of cabbage, chopped
1-26 oz can tomato puree
Fry ground beef with the onion until be
Put all ingredients in a crock pot and co
with a crusty bread. Makes 4 servings.
3
LE SOUP
4
rge onions, sliced
1 T. olive oil
1 T. butter
1/2 lb. linguica
1/2 lb. chorico
1 beef shin bone
4 potatoes, diced and parboiled
Water
In a large pot, saute onions in oil and bu
pan. Sauté linguica and chorico. Remo
add onions, linguica, and chorico; and
C. water. (NOTE: Beef broth may be subs
Add parboiled potatoes, beans, kale, bay
tomatoes. Simmer 1 1/2 - 2 hours. Remove ba
Reheat and serve. Makes 8-10 servings.

Note: Acrobat Pro may automatically add tags when the file is run through OCR.

This example is shown in operation in the working example of generating actual text and the result of performing OCR.

Resources

Resources are for information purposes only, no endorsement implied.

- PDF and Accessibility

Related Techniques

- G140: Separating information and structure from presentation to enable different presentations

Tests

Procedure

1. For each page converted to text using OCR, ensure that the resulting PDF has been converted correctly, using one of the following ways:
    - Read the PDF document with a screen reader or a tool that reads aloud, listening to hear that all text is read correctly and in the correct reading order.
    - Save the document as text and check that the converted text is complete and in the correct reading order.
    - Use a tool that is capable of showing the converted content to open the PDF document and verify that all text was converted and is in the correct reading order.
    - Use a tool that exposes the document through the accessibility API and verify that all text was converted and is in the correct reading order.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF8: Providing definitions for abbreviations via an E entry for a structure element

Applicability

Tagged PDF documents containing abbreviations or acronyms

This technique relates to:

- Success Criterion 3.1.4 (Abbreviations)
  - How to Meet 3.1.4 (Abbreviations)
  - Understanding Success Criterion 3.1.4 (Abbreviations)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF8. Also see PDF Technology Notes.

Description

The objective of this technique is to provide an expansion or definition of an abbreviation for the first occurrence of the abbreviation. For example, a reference to an abbreviation, such as "WCAG," should be available as "Web Content Accessibility Guidelines (WCAG)" on its first occurrence in a document.

This is done by setting expansion text using an /E entry for a structure element, and is normally accomplished using a tool for authoring PDF. A Span structure element is typically used to tag the abbreviation, but the /E entry is valid with any structure element.

This technique is applicable for any abbreviation, including acronyms and initialisms. Note that on the first occurrence of the abbreviation, both the abbreviation and the expansion text must be provided. This will aid recognition of later use of the abbreviation.

PDF documents may be enhanced by providing expansions for abbreviations. In fact, such expansions are required for accessibility to ensure understanding by people who have difficulty decoding words; rely on screen magnification (which may obscure context); have limited memory; or who have difficulty using context to aid understanding.

Examples

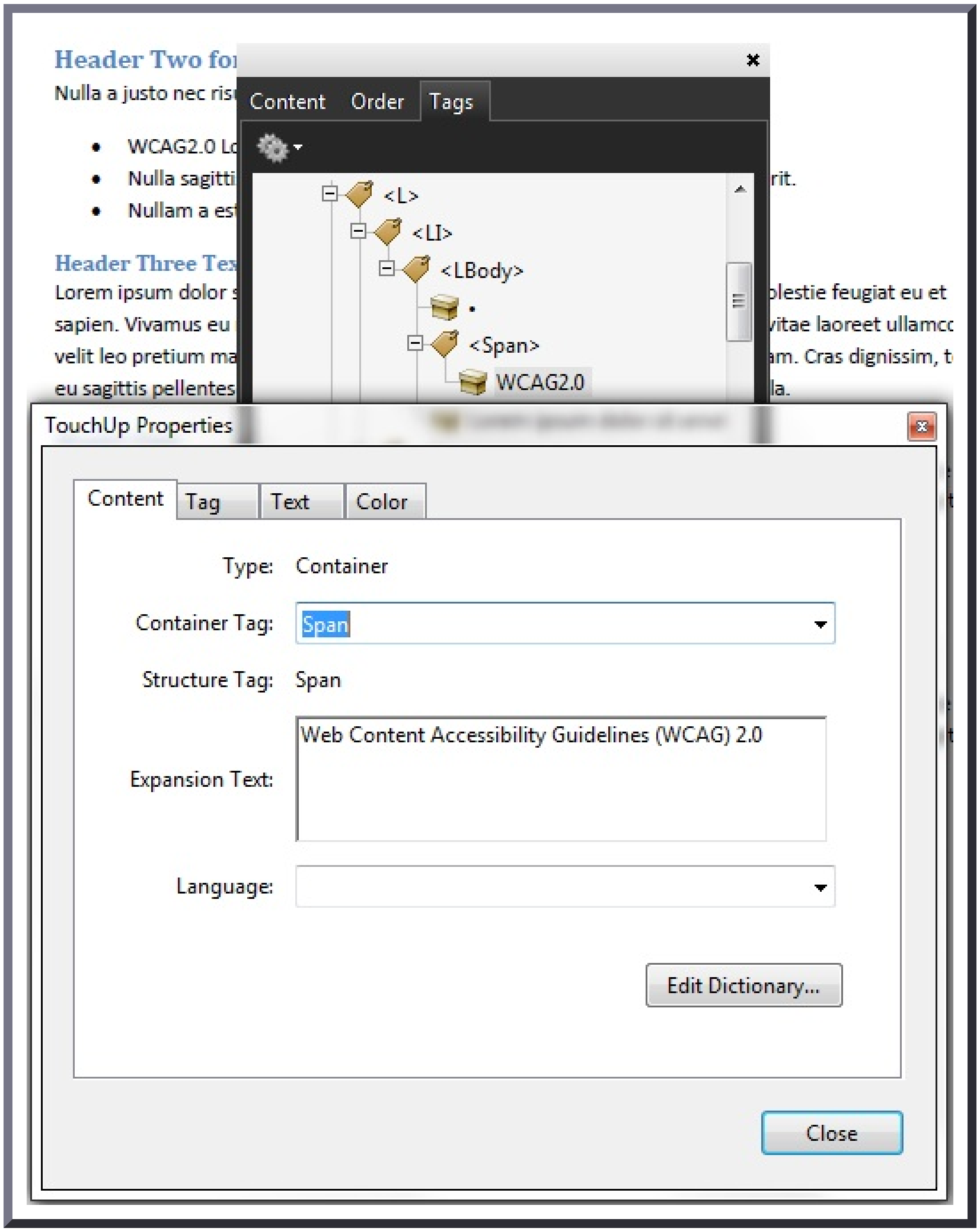
Example 1: Adding an /E entry to an abbreviation using Adobe Acrobat 9 Pro's Tags panel

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

In a tagged PDF document:

1. Select the Tags panel, using Views > Navigation Panes > Tags
2. Select the first instance of the abbreviated text that needs to be expanded. If the selected text is part of a larger tag, access the Tags panel options menu, select Create Tag from Selection, and create a new Span tag. In this example, the text "WCAG2" (within the LBody tag) has been enclosed in a Span tag.
3. In the Tags panel, access the context menu for the spanned text and select Properties... to open the TouchUp Properties dialog for the Span tag.
4. On the Content tab of the TouchUp Properties dialog, enter the expansion text, followed by the originally selected text.

The following image illustrates this technique:



Type: Container

Container Tag: Span

Structure Tag: Span

Expansion Text: Web Content Accessibility Guidelines (WCAG) 2.0

Language:

This example is shown in operation in the working example of Providing definitions for Abbreviations (Word document), working example of Providing definitions for Abbreviations (OpenOffice document), and working example of Providing definitions for Abbreviations (PDF document).

Example 2: Using a /Span structure element with an /E entry to define an abbreviation

The following code fragment illustrates code that is typical for using the /Span structure element to define an abbreviation.

This example uses the sentence "Sugar is commonly sold in 5 lb bags." The abbreviation "lb" is tagged as a /Span structure element with an /E entry (typically accomplished by an authoring tool).

```
1 0 obj                                % structure element
  << /Type /StructElemen
           /S /Span                    % element type
           /P ...                      % Parent in structure hierarchy
           /K << /Type /MCR
                       /Page 2 0 R     % Page containing marked-content sequence
                       /MCID 0         % Marked content identifier for "lb"
              >>
           /E  (pound, lb)
   >>
endobj
```

Example 3: Using a /TH structure element with an /E entry to define an abbreviation

As noted in the Description, the /E entry is valid with any structure element.

The following code fragment illustrates code that is typical for using an /E entry to define an abbreviation.

A table that contains columns for each month uses abbreviations as the values of column headers. The expansion for each abbreviation is provided as the /E entry of the /TH structure element (typically accomplished by an authoring tool).

```
1 0 obj                                 % structure element
  << /Type /StructElemen
           /S /TH                       % element type
           /P ...                       % Parent in structure hierarchy
           /K << /Type /MCR
                       /Page 2 0 R      % Page containing marked-content sequence
                       /MCID 0          % Marked content identifier for "Dec"
              >>
           /E  (December, Dec)
   >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.9.5 (Expansion of Abbreviations and Acronyms) in PDF 1.7 (ISO 32000-1)
- Microsoft Inspect.exe tool
- PDF and Accessibility

Related Techniques

- G102: Providing the expansion or explanation of an abbreviation
- G55: Linking to definitions
- G62: Providing a glossary
- G70: Providing a function to search an online dictionary
- G97: Providing the first use of an abbreviation immediately before or after the expanded form

Tests

Procedure

1. Verify that the first occurrence of abbreviations that require expansion text have /E entries on an enclosing tag by one of the following and that both the abbreviation and the expansion text are provided:
    - In Windows, use Microsoft's Inspect.exe tool, or some other tool that allows inspection of the MSAA interface, to locate the text of the abbreviation in the document tree and ensure that the value of the abbreviation is in the expansion text.
    - In a PDF editor, locate the tag for the text that is the abbreviation, and check that an expansion or definition is provided for each abbreviation in the Expansion Text field in the corresponding tag's properties.
    - Read the PDF document with a screen reader, listening to hear that on the first occurrence, the abbreviation and expansion are read when the screen reader reads the content line-by-line.
    - Use a tool that is capable of showing the /E entry value, such as aDesigner to open the PDF document and view the GUI summary to read the text expansions for abbreviations.
    - Use a tool that exposes the document through the accessibility API and verify that the text expansion of the abbreviation is properly implemented.

Expected Results

- Check #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF9: Providing headings by marking content with heading tags in PDF documents

Applicability

Tagged PDF documents with headings

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
    - How to Meet 1.3.1 (Info and Relationships)
    - Understanding Success Criterion 1.3.1 (Info and Relationships)
- Success Criterion 2.4.1 (Bypass Blocks)
    - How to Meet 2.4.1 (Bypass Blocks)
    - Understanding Success Criterion 2.4.1 (Bypass Blocks)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF9. Also see PDF Technology Notes.

Description

The purpose of this technique is to show how headings in PDF documents can be marked so that they are recognized by assistive technologies. Headings are marked up using the heading elements (H, H1, H2, ... H6) in the structure tree. This is typically accomplished by using a tool for authoring PDF.

Heading markup can be used:

- to indicate start of main content
- to mark up section headings within the main content area
- to demarcate different navigational sections, such as top or main navigation, left or secondary navigation, and footer navigation
- to mark up images (containing text) which have the appearance of headings visually.

Because headings indicate the start of important sections of content, it is possible for assistive technology users to access the list of headings and to jump directly to the appropriate heading and begin reading the content. This ability to "skim" the content through the headings and go directly to content of interest significantly speeds interaction for users who would otherwise access the content slowly.

Examples

Example 1: Adding or modifying tagged headings in PDF documents with Adobe Acrobat 9 Pro

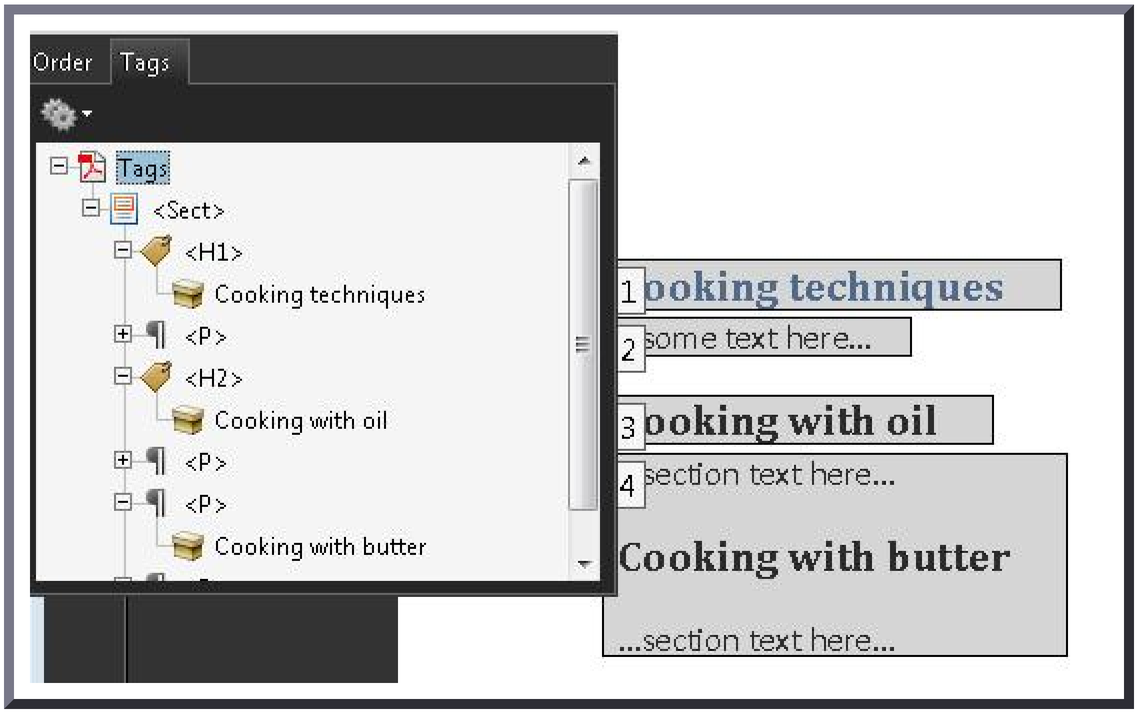
This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Using the Touchup Reading Order tool

One method of adding headings to PDF documents uses the Touchup Reading Order tool:

1. Open the PDF document in Adobe Acrobat Pro
2. Select Advanced > Accessibility > TouchUp Reading Order...
3. Click the Show Order Panel button on the TouchUp Reading Order panel
4. View the tags in the Show Order panel.

The following image shows a PDF document opened in Adobe Acrobat Pro. The Tags panel is open, showing heading text "Cooking techniques" tagged as H1 and "Cooking with oil" tagged as H2. The text "Cooking with butter" should be tagged as H2 but is not.

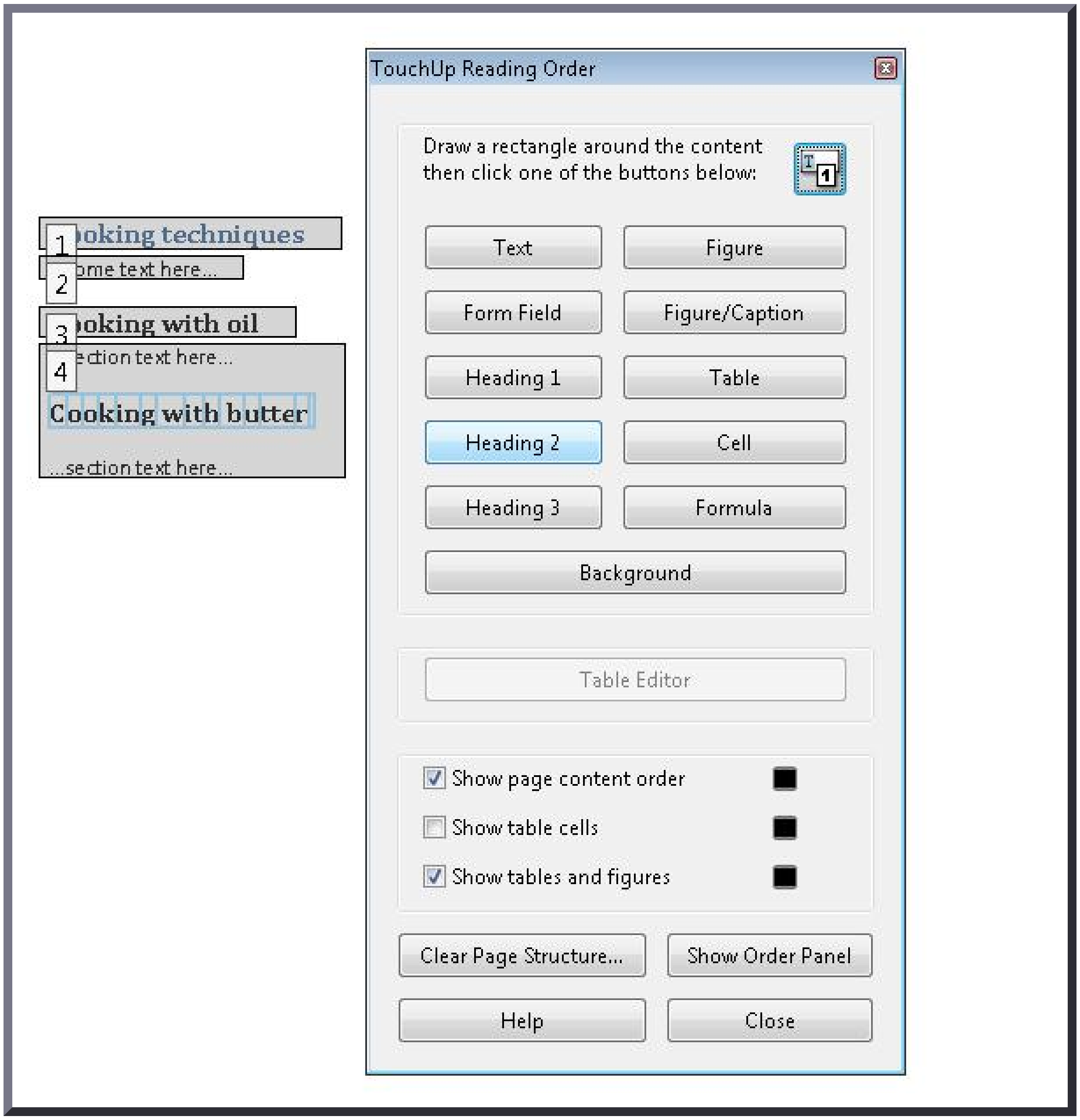


To correct the H2 heading, use the TouchUp Reading Order panel as follows:

1. Left click and drag a selection box over the content you want to tag.

2. Select the Heading 2 tag from the TouchUp Reading Order panel.

The following image shows the PDF document opened in Adobe Acrobat Pro. The TouchUp Reading Order panel is visible. A selection box appears around the text "Cooking with butter," and Heading 2 on the panel is selected.

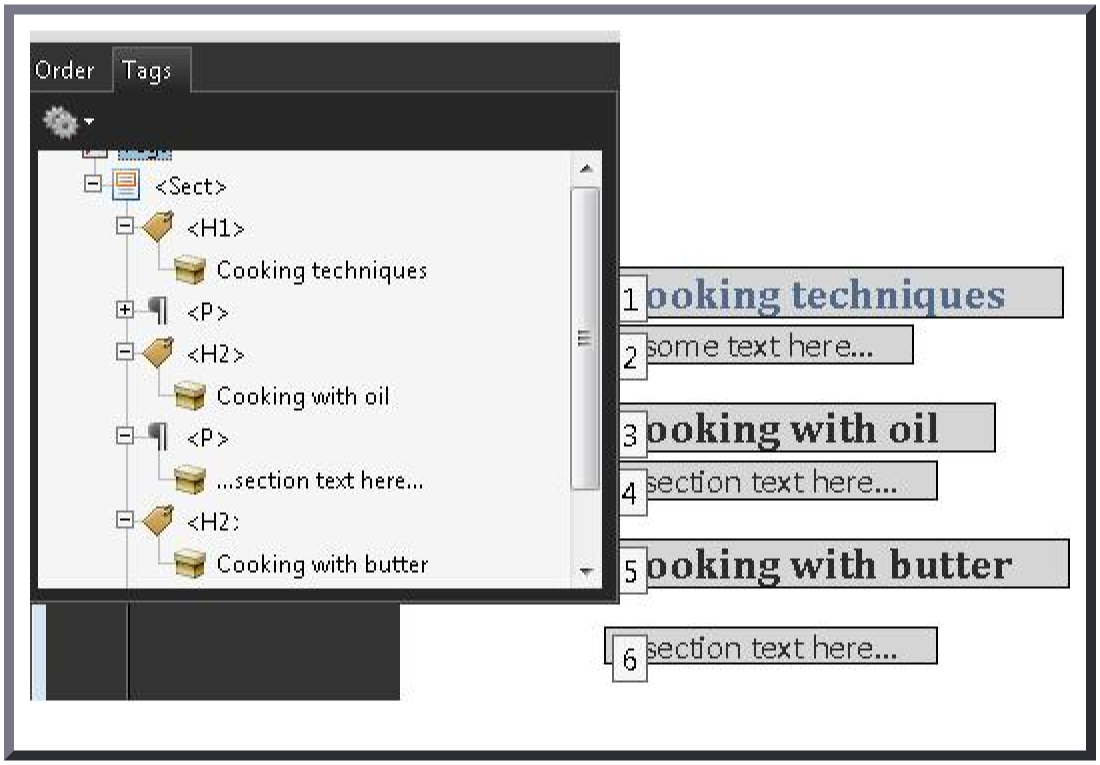


Draw a rectangle around the content then click one of the buttons below:

[✓] Show page content order
[ ] Show table cells
[✓] Show tables and figures

Finally, click the Show Order Panel button on the TouchUp Reading Order panel.

The following image shows the PDF document opened in Adobe Acrobat Pro. The Tags panel is visible, showing that the text "Cooking with butter" is now tagged as H2.

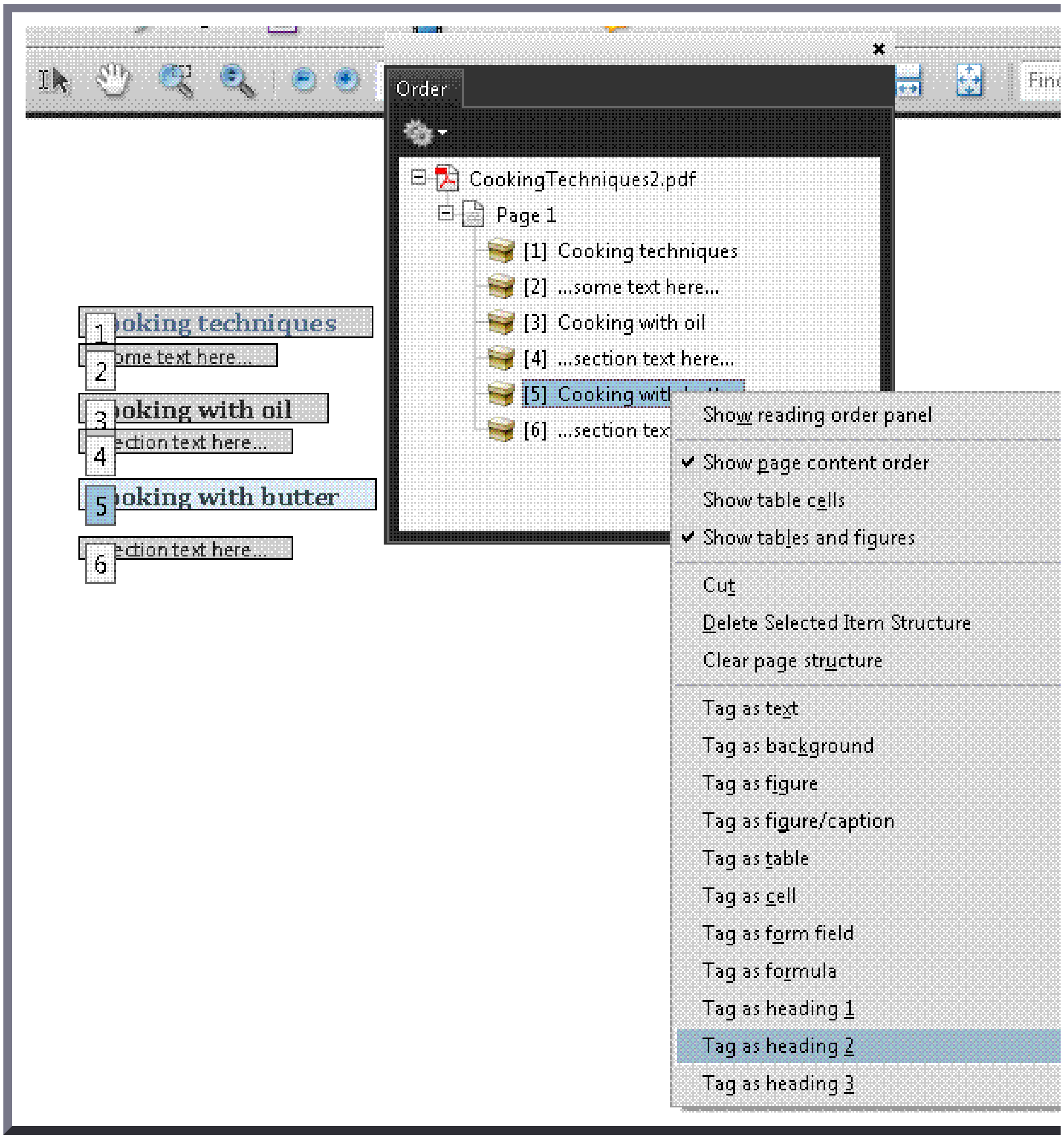


Using the Order and Tags panels

You can also add or change headings as follows:

1. Bring up the Order panel.
2. Access the context menu for the text to be changed or added as a heading.
3. Select the correct heading tag for the text.

The following screenshot shows Order panel and the context menu for the text "Cooking with butter." "Tag as heading 2" is selected in the context menu.



Order

CookingTechniques2.pdf

Page 1

[1] Cooking techniques

[2] ...some text here...

[3] Cooking with oil

[4] ...section text here...

[5] Cooking with

[6] ...section tex

1 oking techniques

2 ome text here...

3 oking with oil

4 ection text here...

5 oking with butter

6 ection text here...

Show reading order panel

✓ Show page content order

Show table cells

✓ Show tables and figures

Cut

Delete Selected Item Structure

Clear page structure

Tag as text

Tag as background

Tag as figure

Tag as figure/caption

Tag as table

Tag as cell

Tag as form field

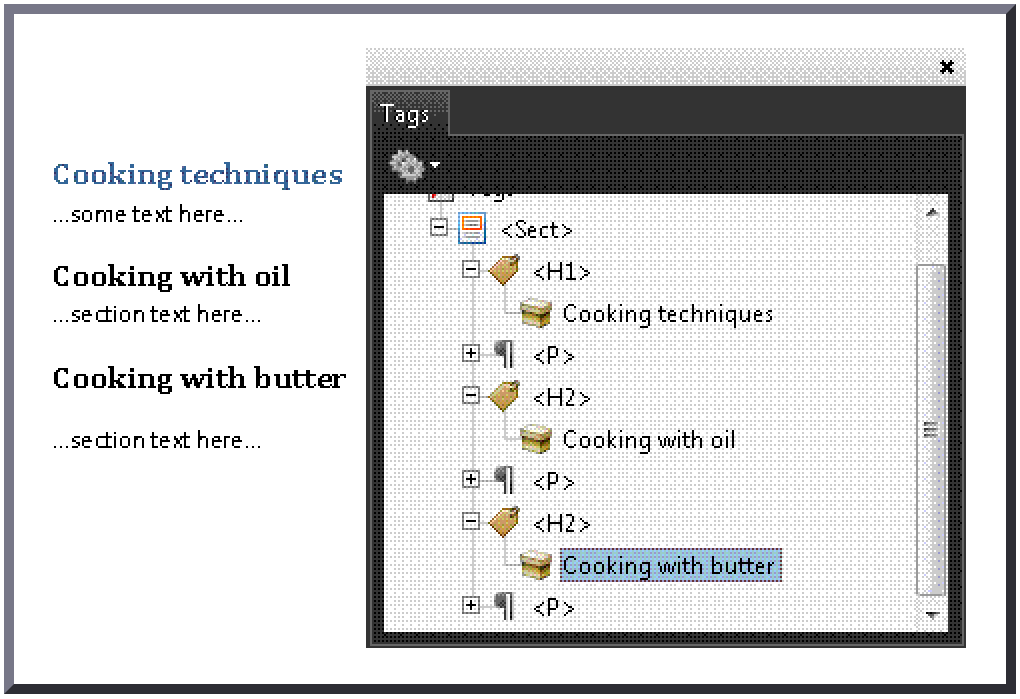
Tag as formula

Tag as heading 1

Tag as heading 2

Tag as heading 3

You can then check that the correct heading is applied by opening the Tags panel, as shown in the following screenshot.


Cooking techniques
...some text here...
Cooking with oil
...section text here...
Cooking with butter
...section text here...

Tags
<Sect>
<H1>
Cooking techniques
<P>
<H2>
Cooking with oil
<P>
<H2>
Cooking with butter
<P>

This example is shown in operation in the working example of adding tagged headings (Word file) and working example of adding tagged headings (PDF file).

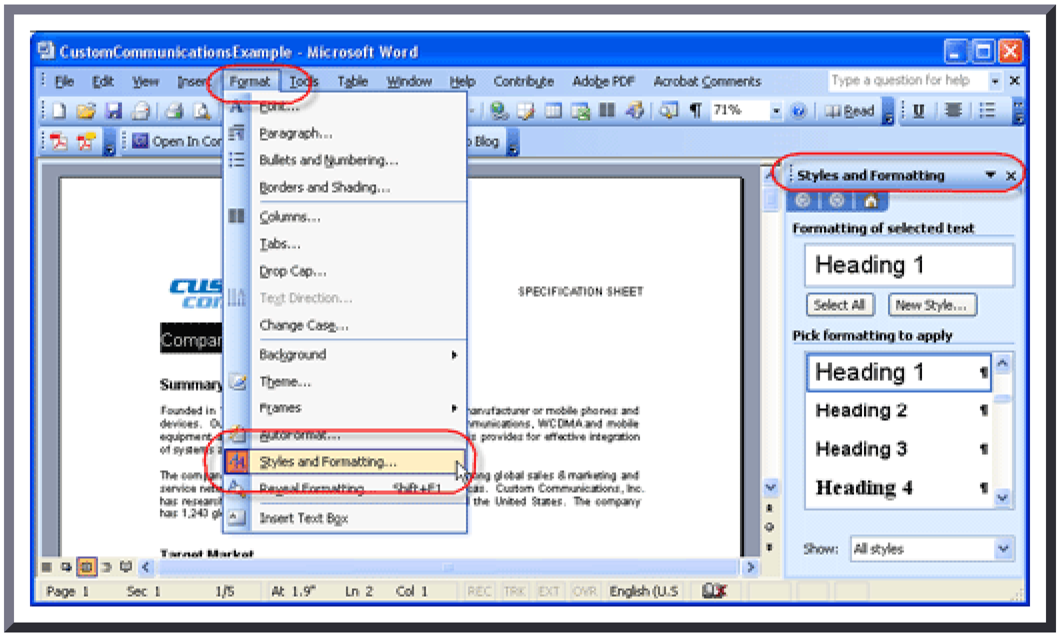
Example 2: Creating documents in Microsoft Word that have correctly tagged headings when converted to PDF

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Use Styles to create heading formats: Heading 1, Heading 2, Heading 3, etc. Make styles progress in a logical manner; e.g., a Heading 2 should come after a Heading 1.

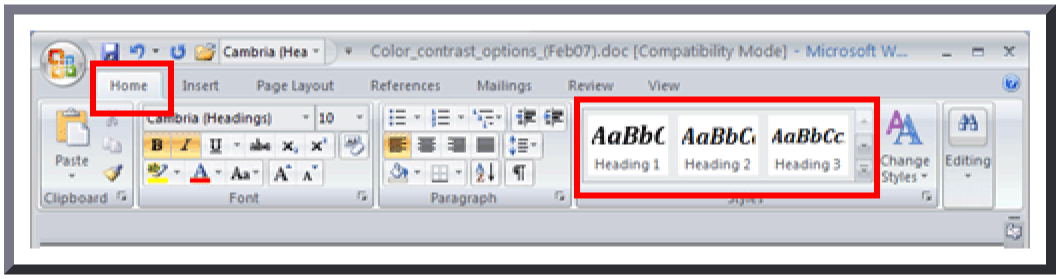
In Microsoft Word 2003

- Select the "Format > Styles and Formatting" menu item to reveal the styles and formatting task pane.
- Use the Heading 1 to Heading 6 styles provided in the "Styles and Formatting" panel.



In Microsoft Word 2007/2010

Select the Home Ribbon in Word 2007/2010 and select the appropriate heading (Heading 1 to Heading 6) from the Styles group.


Color_contrast_options_(Feb07).doc [Compatibility Mode] - Microsoft W...
Home
Insert
Page Layout
References
Mailings
Review
View
Paste
Clipboard
Cambria (Headings)
10
Font
Paragraph
AaBbC
Heading 1
AaBbC
Heading 2
AaBbCc
Heading 3
Styles
Change Styles
Editing

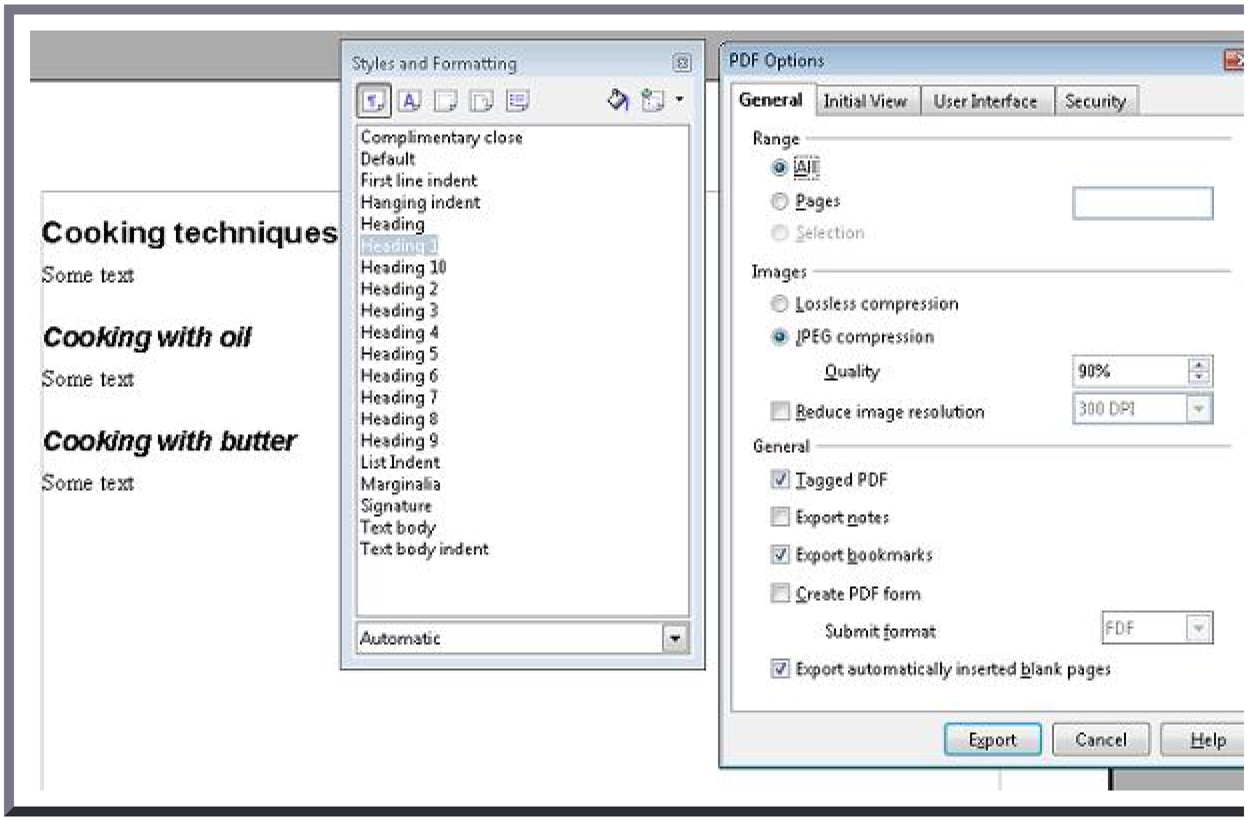
Example 3: Creating documents in OpenOffice.org Writer 2.2 that have correctly tagged headings when converted to PDF

This example is shown with OpenOffice.org Writer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Use Styles to create heading formats: Heading 1, Heading 2, Heading 3, etc. Make styles progress in a logical manner; e.g., a Heading 2 should come after a Heading 1.

Export to PDF as follows:

1. From the File menu, select Export as PDF...
2. The first time you export as PDF, an Options Dialog appears.
3. Select Tagged PDF, then select Export.



Cooking techniques

Some text

Cooking with oil

Some text

Cooking with butter

Some text

Complimentary close
Default
First line indent
Hanging indent
Heading
Heading 1
Heading 10
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6
Heading 7
Heading 8
Heading 9
List Indent
Marginalia
Signature
Text body
Text body indent

Automatic

Range
[✓ All]
Pages
Selection

Images
Lossless compression
[✓ JPEG compression]
Quality 90%
Reduce image resolution 300 DPI

General
[✓ Tagged PDF]
Export notes
[✓ Export bookmarks]
Create PDF form
Submit format FDF
[✓ Export automatically inserted blank pages]

Example 4: Marking up headings using /Hn elements

Headings within PDF documents can be marked up using /H*n* elements elements in the structure tree, where *n* is numeral 1 through 6 (for example /H1, /H2, etc.).

The following code fragment illustrates code that is typical for using the /H*n* elements elements to mark content. Note that /H1 has been role-mapped to /Head1 in this example. This is typically accomplished by an authoring tool.

```
0 obj% Document catalog
  << /Type /Catalog
     /Pages 100 0 R                 % Page tree
     /StructTreeRoot 300 0 R        % Structure tree root
  >>
endobj
 ...
300 0 obj% Structure tree root
  << /Type /StructTreeRoot
     /K [ 301 0 R                   % Two children: a chapter
        304 0 R                     % and a paragraph
        ]
     /RoleMap << /Chap /Sect        % Mapping to standard structure types
                 /Head1 /H
                 /Para /P
              >>
    /ClassMap << /Normal 305 0 R >>  % Class map containing one attribute class
    /ParentTree 400 0 R             % Number tree for parent elements
    /ParentTreeNextKey 2            % Next key to use in parent tree
    /IDTree 403 0 R                 % Name tree for element identifiers
  >>
endobj
301 0 obj                           % Structure element for a chapter
  << /Type /StructElem
     /S /Chap
     /ID (Chap1)                    % Element identifier
     /T (Chapter 1)                 % Human-readable title
     /P 300 0 R                     % Parent is the structure tree root
     /K [ 302 0 R                   % Two children: a section head
          303 0 R                   % and a paragraph
        ]
  >>
endobj
302 0 obj                           % Structure element for a section head
  << /Type /StructElem
     /S /Head1
     /ID (Sec1.1)                   % Element identifier
     /T (Section 1.1)               % Human-readable title
     /P 301 0 R                      % Parent is the chapter
     /Pg 101 1 R                     % Page containing content items
     /A << /O /Layout                % Attribute owned by Layout
           /SpaceAfter 25
           /SpaceBefore 0
           /TextIndent 12.5
        >>
    /K 0                             % Marked-content sequence 0
  >>
endobj
...
```

Within marked content containers, headings can be marked up using /Head*n* elements as follows for a first-level heading in a PDF document:

```
BT                              % Start of text object
  /Head1 <</MCID 0 >>           % Start of marked-content sequence
     BDC
        ...
        (This is a first level heading. Hello world: ) Tj
        ...
     EMC                        % End of marked-content sequence
     ...
ET                              % End of text object
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8.4.3.2 (Paragraphlike Elements) in PDF 1.7 (ISO 32000-1)
- PDF Accessibility Documentation:headings
- PDF and Accessibility

Related Techniques

- G141: Organizing a page using headings

Tests

Procedure

1. For all PDF content that is divided into separate sections, use one of the following to verify that headings are tagged correctly:
    - Read the PDF document with a screen reader, listening to hear that the list of headings is announced correctly.
    - Using a PDF editor, make sure the headings are tagged correctly.
    - Use a tool that is capable of showing the /Head*n* entries to open the PDF document and verify that headings are tagged correctly.
    - Use a tool that exposes the document through the accessibility API and verify that the headings are tagged correctly.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF10: Providing labels for interactive form controls in PDF documents

Applicability

- Tagged PDF documents with forms.
- PDF forms created using Adobe LiveCycle Designer.

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
    - How to Meet 1.3.1 (Info and Relationships)
    - Understanding Success Criterion 1.3.1 (Info and Relationships)

- Success Criterion 3.3.2 (Labels or Instructions)
  - How to Meet 3.3.2 (Labels or Instructions)
  - Understanding Success Criterion 3.3.2 (Labels or Instructions)
- Success Criterion 4.1.2 (Name, Role, Value)
  - How to Meet 4.1.2 (Name, Role, Value)
  - Understanding Success Criterion 4.1.2 (Name, Role, Value)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF10. Also see PDF Technology Notes.

Description

The objective of this technique is to ensure that users of assistive technology are able to perceive form control labels and understand how form controls are used.

Form controls allow users to interact with a PDF document by filling in information or indicating choices which can then be submitted for processing. Assistive technology users must be able to recognize and understand the form fields, make selections, and provide input to complete the forms, and submit the form, just as sighted users can. Understandable labels that convey the purpose of each form control are essential to form accessibility.

Form inputs generally have labels and instructions to help users understand what information is required and how to fill in the form. Unless these labels are programmatically associated with the relevant fields, assistive technology might not be able to associate them correctly, and thus users might not understand how to complete the form.

Using Adobe Acrobat Pro with documents with interactive forms, you can make sure that the forms are accessible and usable by making sure that programmatically associated labels that convey the purpose of the fields are provided.

The heuristics used by assistive technology will sometimes use the text label if a programmatically associated label cannot be found. The TU entry (which is the tooltip) of the field dictionary is the programmatically associated label (see Example 3 below and Table 220 in PDF 1.7 (ISO 32000-1)). Therefore, add a tooltip to each field to provide a label that assistive technology can interpret.

Placement rules

The table below lists the placement rules governing where Adobe LiveCycle positions labels by default. Note that these rules assume left-to-right text directionality. If your form requires different positioning (e.g., to accommodate PDF documents in languages that use right-to-left text directionality), see *Repositioning form labels* in Example 2 below. In general, authors should review label positioning to make sure it meets the requirements of their particular form.

Control Type	**LiveCycle Placement Rules**
Text input (including date/time and password fields)	Default placement for the label is to the left of the control. If this is not possible, LiveCycle will attempt to place it immediately above the control.
Checkbox	Default placement for the label is to the right of the check box.
Radio button group	Default placement for the label for each individual radio button is to the right of the button. Create a visible caption for the radio button group by creating static text and placing it to the left of or above the group. (See *Labeling radio buttons* below.)
Combo box	Default placement for the label is to the left of the drop-down list. If this is not possible, LiveCycle will attempt to place it immediately above the control..
List box	Default placement for the label is above the list box.
Button	LiveCycle automatically places the label on the button; it does not have to be positioned manually. Ensure that the button's purpose is properly described in the label text.

Examples

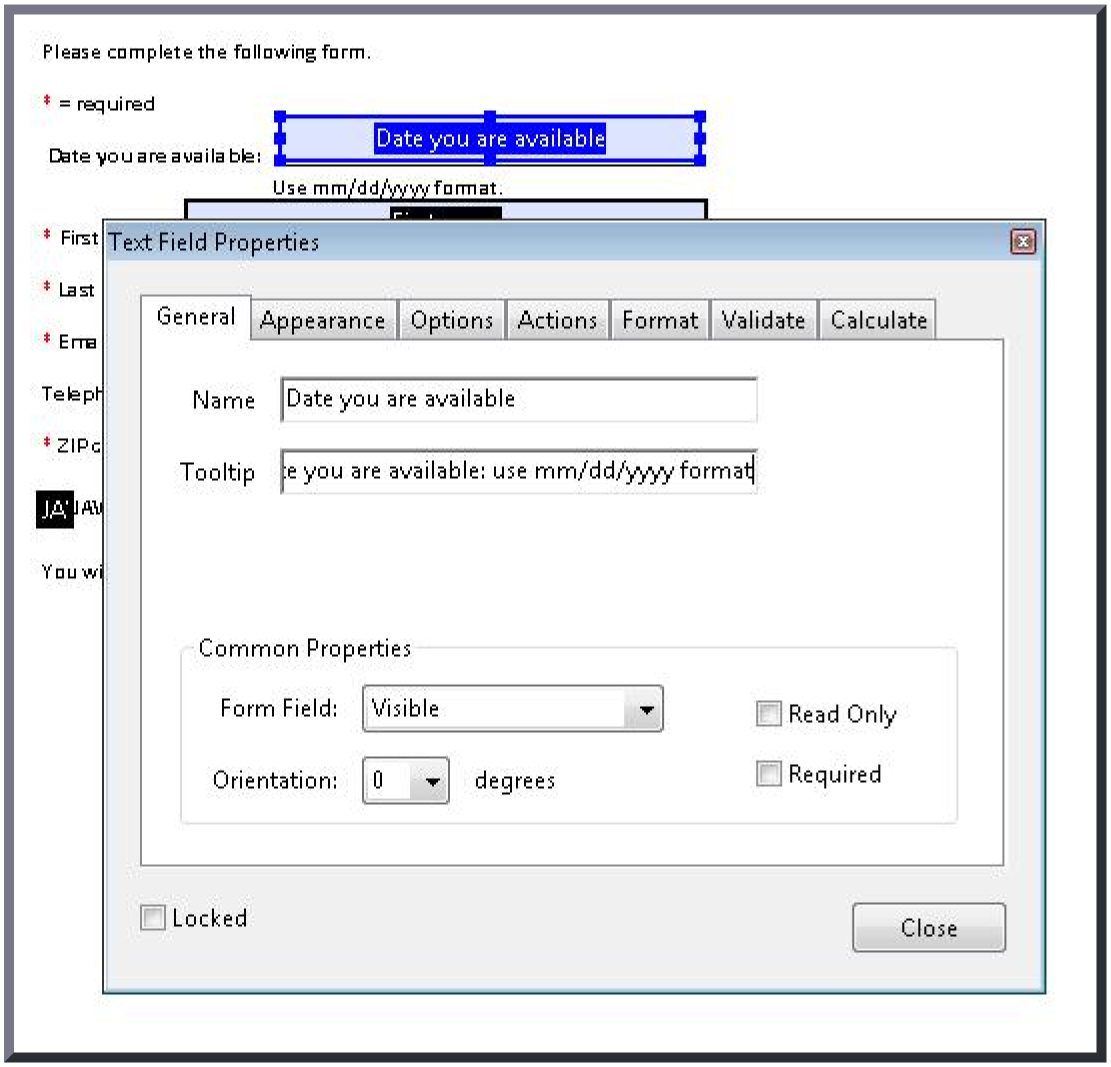
Example 1: Providing labels using the Forms tool in Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

As noted in the Description, text labels added in an authoring tool and then converted to PDF might be visually associated with the fields but are not programmatically associated, and you should provide a tooltip.

1. In the Forms menu, select Add or Edit Fields...
2. For the field you want to edit, access the context menu and select the Properties dialog.
3. In the General tab of the Properties dialog, type a description for the form field in the Tooltip field.
4. Repeat for all form fields.

The following image shows the Properties dialog with a description in the Tooltip field.


Please complete the following form.

* = required

Date you are available: Date you are available

Use mm/dd/yyyy format.

Text Field Properties

Name: Date you are available

Tooltip: te you are available: use mm/dd/yyyy format

Common Properties

Form Field: Visible

Orientation: 0 degrees

☐ Read Only

☐ Required

☐ Locked

This example is shown in operation in the working example of providing labels using the forms tool.

Example 2: Providing labels to form controls in Adobe LiveCycle Designer ES 8.2.1

This example is shown with Adobe LiveCycle Designer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

LiveCycle Designer provides several options for associating descriptive text and labels with form elements.

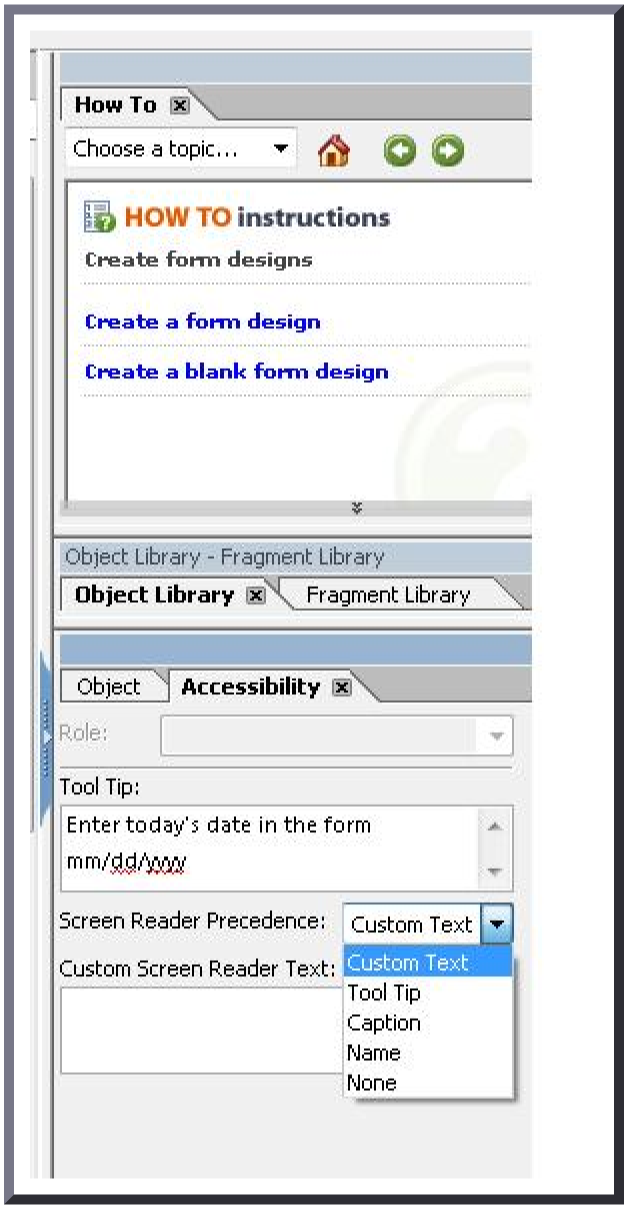
For sighted or low-vision users, it is important to properly position the label adjacent to the control. For screen reader users, you should also ensure that the label is programmatically associated with the form control and that sufficient information is provided so that screen reader users can readily complete and submit the form.

This example is shown in operation in the working example of providing labels in LiveCycle Designer.

Specifying accessible label text using the accessibility palette

In LiveCycle Designer, create or import a form. Then:

1. Enable the palette by selecting Window > Accessibility or by pressing shift + F6.
2. The palette appears in LiveCycle Designer's right-hand panel.
3. Select an object in your form. The palette shows the object's accessibility properties.



Tool Tip:

Enter today's date in the form mm/dd/yyyy

Screen Reader Precedence: Custom Text

Custom Screen Reader Text:

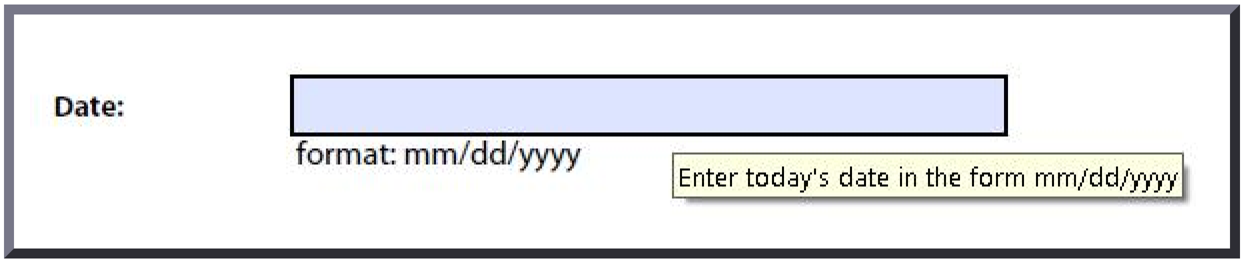
The label that a screen reader uses does not necessarily have to be the same as the visual caption. In some cases, you may want to provide more information about a form element's purpose.

To specify what text should be announced by the screen reader for a particular object, you can use the Accessibility Palette's Screen Reader Precedence drop down list. Text is announced in the order shown in the list: custom text, tool tip, caption, and name.

Depending on the complexity and difficulty of your form, you must decide which option best suits the requirements for your form.

By default, a screen reader searches for an object's text in order shown in the image. Once descriptive text has been found for a control, the search stops.

The image below shows an example of a text field with a visual caption that might be unclear for screen reader users. One of the fields has a caption of "Date" but screen reader users may want to know the preferred date format (shown as screen text). So this text is provided in the tooltip. Because a tooltip has a higher precedence than the visual caption, the screen reader uses the tooltip.



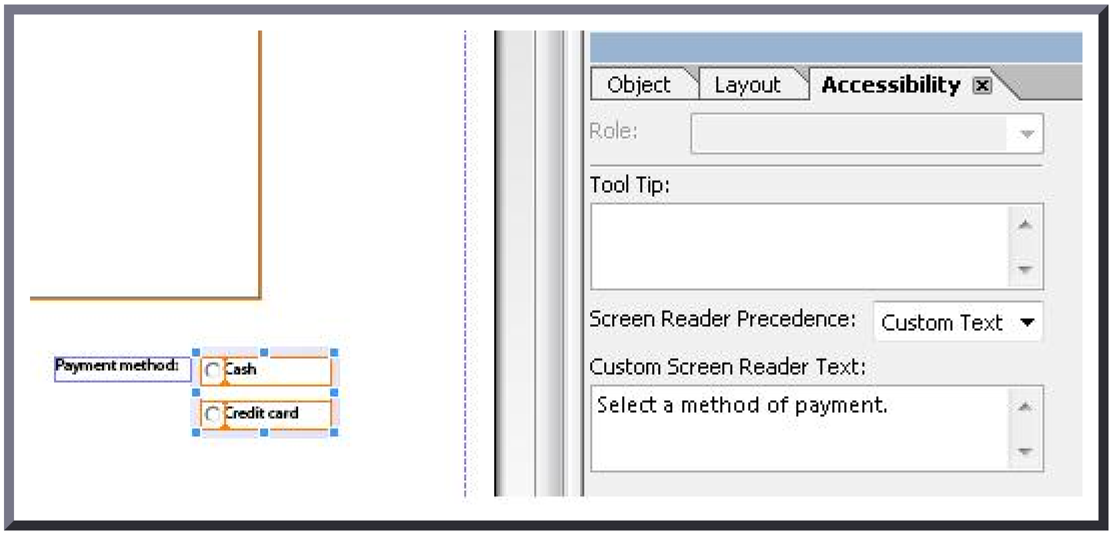
Labeling radio buttons

When a screen reader user tabs into a radio button, the screen reader needs to announce two items:

- A general description of the purpose of the *group* of buttons
- A meaningful description for the purpose of *each* radio button

To make radio buttons accessible:

1. In the Hierarchy palette, select the radio button group.
2. Select the Accessibility palette and in the Custom Screen Reader Text box, type the speak text for the group. For example, type "Select a method of payment."
3. In the Hierarchy palette, select the first radio button in the group.
4. In the Object palette, select the Field tab. In the Item area, select the item and type a meaningful value for the selected radio button. For example, type "Cash."
5. Repeat steps 3 and 4 for each radio button in the group.



Role:

Tool Tip:

Screen Reader Precedence: Custom Text

Custom Screen Reader Text: Select a method of payment.

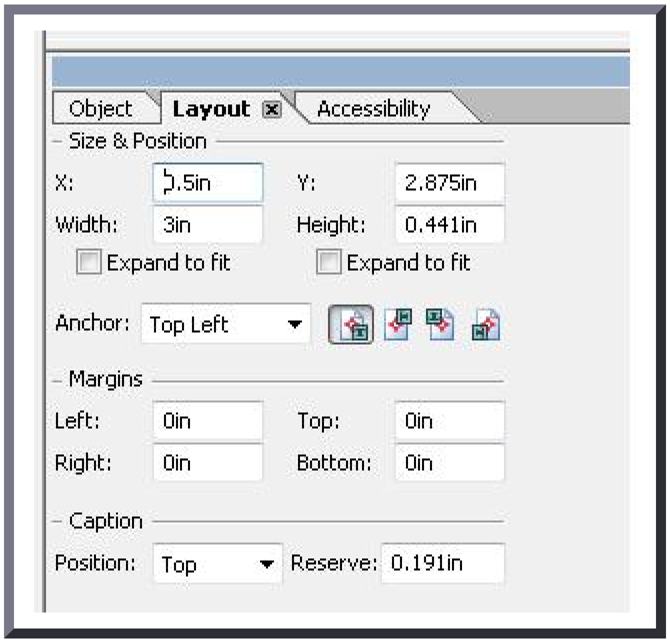
Repositioning form labels

The placement of a caption, or label, is important because users expect them to be found at a particular location adjacent to the control. For screen magnification users this is even more important, as they might not be able to view both the control and the label at the same time.

When you create an object, Adobe LiveCycle Designer automatically positions the label as specified by the control type (see the table in the Description above). For example, for a text field, the label is positioned to the left of the control.

If you need to change the position of the label text (for example, to accommodate right-to-left text directionality):

1. Select the object by moving the focus to it.
2. In the Layout palette, under Caption at the bottom of the palette, select the position of your object from the Position drop-down list.



Size & Position

X:	5.5in	Y:	2.875in
Width:	3in	Height:	0.441in
[ ] Expand to fit		[ ] Expand to fit	

Anchor: Top Left

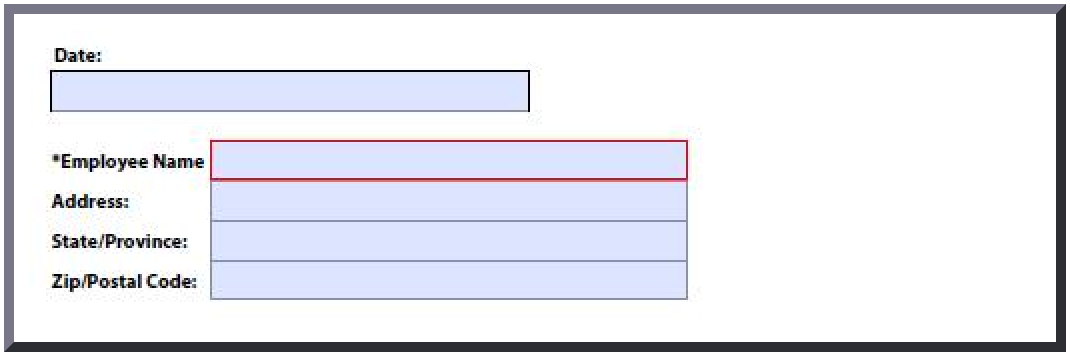
Margins

Left:	0in	Top:	0in
Right:	0in	Bottom:	0in

Caption

Position: Top Reserve: 0.191in

The resulting repositioned label is shown below. The label for the Date text field has been moved from the left of the field to the line above the field.


Date:

*Employee Name

Address:

State/Province:

Zip/Postal Code:

Example 3: Adding a tooltip to interactive form controls

The following code fragment illustrates the use of the TU entry to provide a tooltip (or programmatically associated text label) for a form field. This is typically accomplished by an authoring tool.

```
<< /AP -dict-
   /DA /Helv  0 Tf 0 g
   /DR -dict-
   /F 0x4
   /FT Tx              % FT key set to Tx for Text Field
   /P -dict-
   /Rect -array-
   /StructParent 0x1
   /Subtype Widget
   /T Date you are available   % Partial field name Date
   /TU Date you are available: use MM/DD/YYYY format % TU tool tip value serves as short des
   /Type Annot
   /V Pat Jones
>>
...
<Start Stream>
 BT
  /P <</MCID 0 >>BDC
  /CS0 cs 0  scn
  /TT0 1 Tf
    -0.001 Tc 0.003 Tw 11.04 0 0 11.04 72 709.56 Tm
    [(P)-6(le)-3(as)10(e)-3( )11(P)-6(rin)2(t)-3( Y)8(o)-7(u)2(r N)4(a)11(m)-6(e)]TJ
  0 Tc 0 Tw 9.533 0 Td
  ( )Tj
  -0.004 Tc 0.004 Tw 0.217 0 Td
  [(\()-5(R)-4(e)5(q)-1(u)-1(i)-3(r)-3(e)-6(d)-1(\))]TJ
 EMC
  /P <</MCID 1 >>BDC
  0 Tc 0 Tw 4.283 0 Td
  [( )-2( )]TJ
   EMC
   /ArtifactSpan <</MCID 2 >>BDC
   0.002 Tc -0.002 Tw 0.456 0 Td
  [(__)11(___)11(___)11(___)11(___)11(_)11(____)11(___)11(___)11(__)]TJ
  0 Tc 0 Tw 13.391 0 Td
  ( )Tj
  EMC
 ET
<End Stream>
```

Resources

Resources are for information purposes only, no endorsement implied.

- PDF 1.7 (ISO 32000-1)
- Adobe XML Forms Architecture (XFA)
- PDF and Accessibility

Related Techniques

- G131: Providing descriptive labels
- G162: Positioning labels to maximize predictability of relationships
- PDF23: Providing interactive form controls in PDF documents
- PDF5: Indicating required form controls in PDF forms
- PDF22: Indicating when user input falls outside the required format or values in PDF forms

Tests

Procedure

1. For each form control, verify visually that the label is positioned correctly in relation to the control.
2. For each form control, verify that the name is programmatically associated with the control by one of the following:
   - Open the PDF document with a tool that is capable of showing the name associated with the control and verify that the name is associated correctly with the control.
   - Use a tool that exposes the document through the accessibility API, and verify that the name is associated correctly with the control.

Expected Results

- #1 and #2 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF11: Providing links and link text using the Link annotation and the /Link structure element in PDF documents

Applicability

PDF documents that contain links

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
  - How to Meet 1.3.1 (Info and Relationships)
  - Understanding Success Criterion 1.3.1 (Info and Relationships)
- Success Criterion 2.1.1 (Keyboard)
  - How to Meet 2.1.1 (Keyboard)
  - Understanding Success Criterion 2.1.1 (Keyboard)
- Success Criterion 2.1.3 (Keyboard (No Exception))
  - How to Meet 2.1.3 (Keyboard (No Exception))
  - Understanding Success Criterion 2.1.3 (Keyboard (No Exception))
- Success Criterion 2.4.4 (Link Purpose (In Context))
  - How to Meet 2.4.4 (Link Purpose (In Context))
  - Understanding Success Criterion 2.4.4 (Link Purpose (In Context))

    Note: This technique must be combined with other techniques to meet SC 2.4.4. See Understanding SC 2.4.4 for details.
- Success Criterion 2.4.9 (Link Purpose (Link Only))
  - How to Meet 2.4.9 (Link Purpose (Link Only))

  - Understanding Success Criterion 2.4.9 (Link Purpose (Link Only))

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF11. Also see PDF Technology Notes.

Description

The purpose of this technique is to show how link text in PDF documents can be marked up to be recognizable by keyboard and assistive technology users. That is, the link information is programmatically available to user agents so that links are recognizable when presented in a different format. This is typically accomplished by using a tool for authoring PDF.

Links in PDF documents are represented by a Link tag and objects in its sub-tree, consisting of a link object reference (or Link annotation) and one or more text objects. The text object or objects inside the Link tag are used by assistive technologies to provide a name for the link.

The simplest way to provide links that comply with the WCAG success criteria is to create them when authoring the document, before conversion to PDF.

However, in some cases, it may not be possible to create the links using the original authoring tool. In these cases, Adobe Acrobat Pro can be used to create the link. But, because the tooltip created using the Link dialog in Adobe Acrobat Pro is not accessible to screen readers, be sure that the link text or the link context makes the purpose clear.

In all cases, link purpose should be made clear as described in the general techniques:

- G53: Identifying the purpose of a link using link text combined with the text of the enclosing sentence
- G91: Providing link text that describes the purpose of a link

Examples

Example 1: Creating a hyperlink in Microsoft Word 2007 before conversion to PDF

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

To create a hyperlink in Microsoft Word, first locate the item (e.g., web page) to link to. Then:

1. Either
   - Select Insert on the ribbon and select Hyperlink in the Links tools
   - Or, use the CTRL+K keyboard shortcut
2. On the Insert Hyperlink dialog, add the link destination and link text.
3. Save the file as tagged PDF. (See the PDF Technology Notes.)

Example 2: Creating a hyperlink in OpenOffice.org Writer 2.2 before conversion to PDF

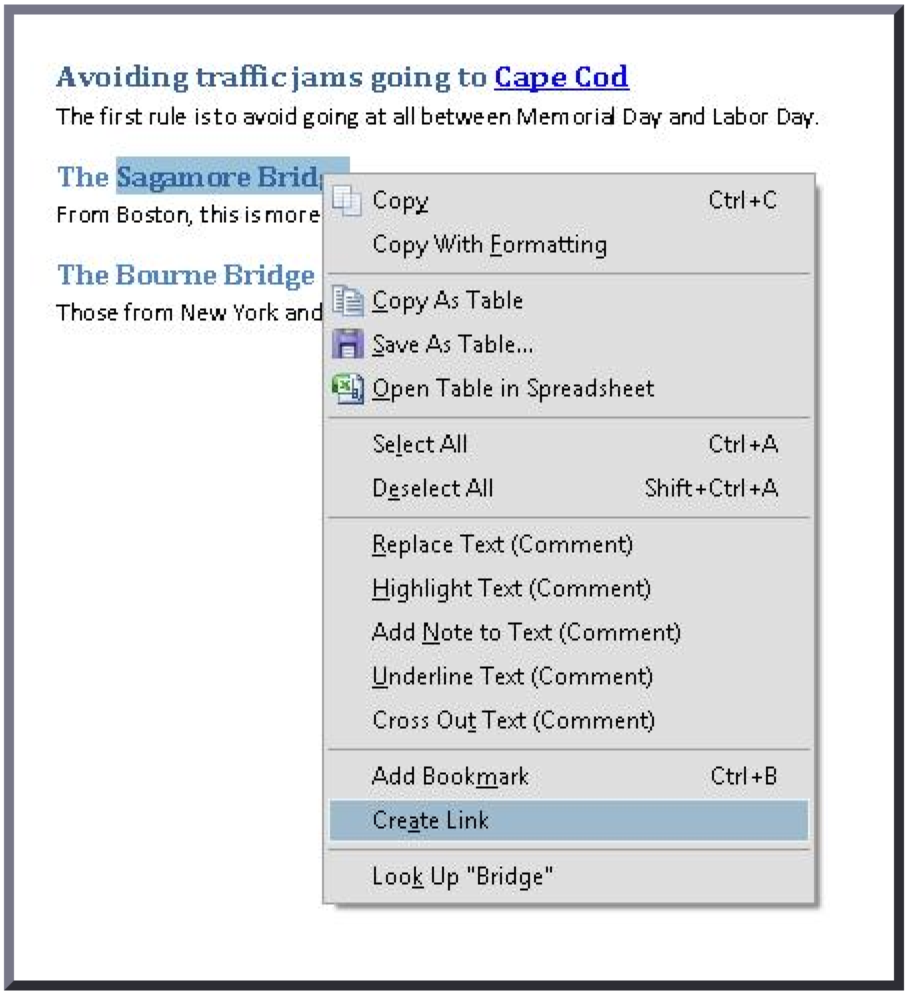
This example is shown with OpenOffice.org Writer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. On the Insert menu, select Hyperlink.
2. In the Hyperlink dialog, insert the target URI in the Target field under Hyperlink Type.
3. Insert the link text in the Text field under Further Settings. (You can also select the link text from the document text before bringing up the dialog. The Text field will be filled in with the selected text.)
4. Save the file as tagged PDF. (See the PDF Technology Notes.)

Example 3: Creating a hyperlink using the Create Link dialog in Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Select the text that will become the link text.
2. Access the context menu and select Create Link.



Avoiding traffic jams going to Cape Cod

The first rule is to avoid going at all between Memorial Day and Labor Day.

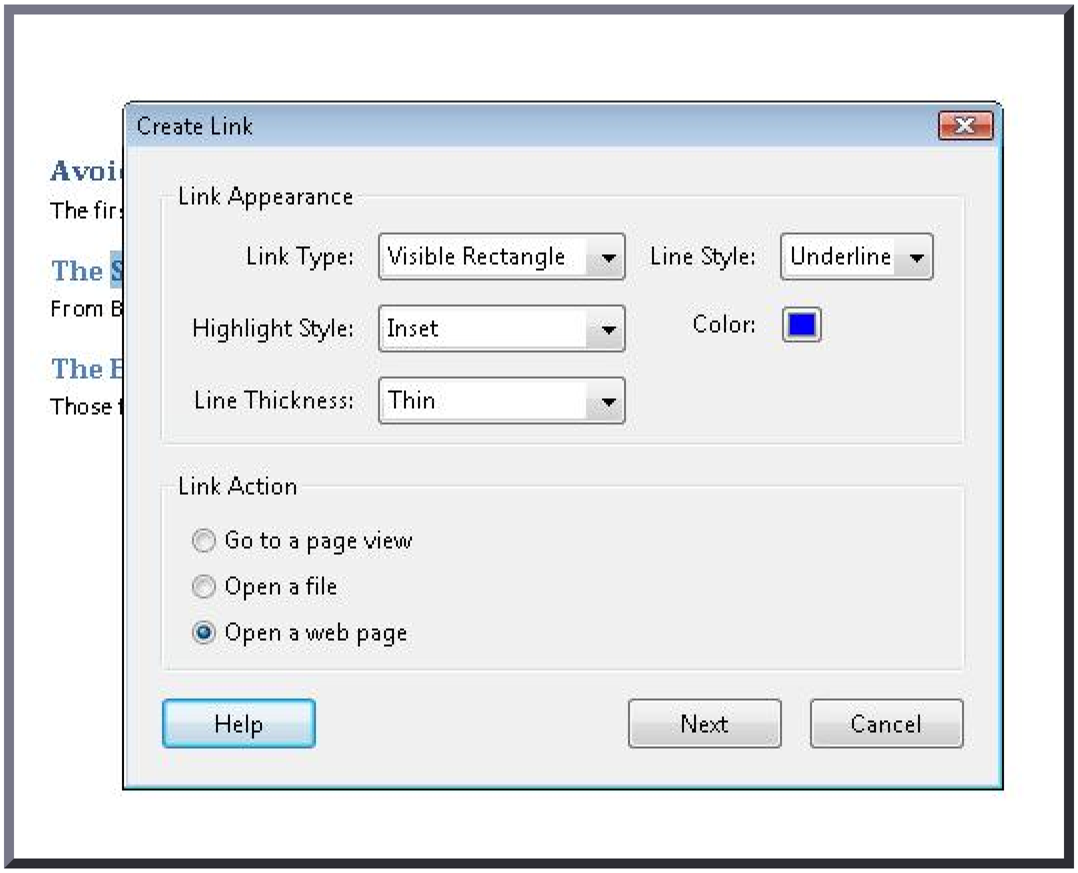
The Sagamore Brid

From Boston, this is more

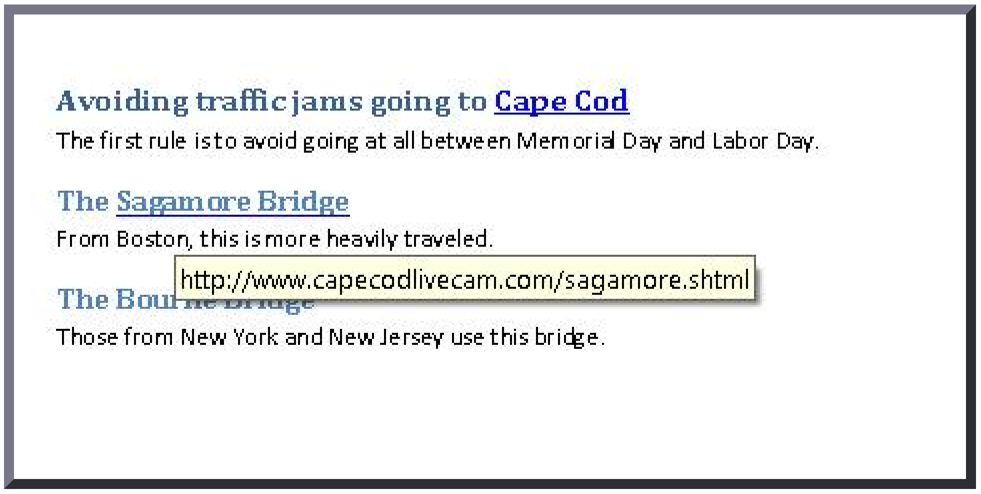
The Bourne Bridge

Those from New York and

3. Follow the instructions in the Create Link dialog to specify the link appearance, as shown below.



Then select Next and specify the URI. The image below shows the resulting hyperlink and tooltip.


Avoiding traffic jams going to Cape Cod

The first rule is to avoid going at all between Memorial Day and Labor Day.

The Sagamore Bridge

From Boston, this is more heavily traveled.

http://www.capecodlivecam.com/sagamore.shtml

The Bourne Bridge

Those from New York and New Jersey use this bridge.

This example is shown in operation in the working example of creating a hyperlink in PDF.

Example 4: Marking up link text using a /Link structure element

Link annotations in PDF documents are associated with a geometric region of a page rather than a particular object in a content stream. For this reason, link annotations alone are not useful for users with visual impairments, or to applications that must determine which content can be activated to invoke a hypertext link.

Tagged PDF /Link elements use PDF's logical structure to establish the association between content items and link annotations, providing functionality comparable to HTML hypertext links.

In HTML, the following example produces text containing a hypertext link:

```
Here is some text <a href="http://www.w3.org/WAI/"> with a link </a> inside.
```

In PDF the page must be painted first and then a link annotation placed over the area where the object action will occur.

The following code fragment shows PDF equivalent to the HTML above; it uses link text displayed in blue and underlined. A second code fragment follows, indicating the associated logical structure hierarchy. This is typically accomplished by an authoring tool.

```
/P <</MCID 0>>                                              %Marked Content Sequence 0 (p
 BDC                                                        %Begin marked content sequenc
  BT                                                        %Begin text object
   /F1 11.04 Tf                                             %set text font and size
   1 0 0 1 72.024 709.54 Tm                                 %set text matrix
   0 g                                                      %set non stroking color to bl
   0 G                                                      %set stroke color to black
  [(H)3(ere )-4(is s)10(o)5(m)-4(e)9( t)-3(e)9(xt)-3( )] TJ  %Show text preceding the link
  ET                                                        %end text object
 EMC                                                        %end marked content sequence

/Span <</MCID 1>>                                           %Marked Content Sequence 1 (u
 BDC                                                        %Begin marked content sequenc
  BT                                                        %Begin text object
   1 0 0 1 152.42 709.54 Tm                                 %set text matrix
   0 0 1 rg                                                 %set non-stroking color to bl
   0 0 1 RG                                                 %set stroke color to blue
   [(with a )-2(li)3(n)14(k)] TJ                            %Show link text " with a link
  ET                                                        %end text object
   0 0 1 rg                                                 %set stroke color to blue
   152.42 707.62 45.984 0.72 re                             %rectangle operator - target
   f*                                                       %fill the path using the ever
 EMC                                                        %end marked content sequence
```

```
/P <</MCID 2>>                                              %Marked Content Sequence 2 (p
 BDC                                                        %Begin marked content sequenc
  BT                                                        %begin text object
   1 0 0 1 198.41 709.54 Tm                                 %set text matrix
   0 g                                                      %set non stroking color to bl
   0 G                                                      %set stroke color to black
   [( )] TJ                                                 %empty text string showing wh
  ET                                                        %end text object
  BT                                                        %begin text object
   1 0 0 1 200.93 709.54 Tm                                 %set text matrix
   [(in)5(sid)5(e.)] TJ                                     %show text following the link
  ET                                                        %end text
  BT                                                        %begin text object
   1 0 0 1 229.97 709.54 Tm                                 %set text matrix
   [( )] TJ                                                 %empty text string showing wh
  ET                                                        %end text object
 EMC                                                        %end marked content sequence
```

The following code fragment is an excerpt from the logical structure that establishes the association between the content items and the link annotation:

```
11 0 obj                                            %Object ID 11, generation   0, obj ke
 <</K[1                                             %immediate child of the structure tre
  <<
   /Obj 26 0 R                                      %reference to Object 26
   /Type/OBJR                                       %this object describes an indirect ob
  >>]
   /P 12 0 R
   /Pg 17 0 R
   /S/Link
 >>
endobj

26 0 obj                                            %object ID 26 which is referenced by
 <</A 31 0 R
  /BS
  <</S/S
    /Type/Border
    /W 0
  >>
  /Border[0 0 0]                                    %a colorless border
  /H/I
  /Rect[150.128 694.558 200.551 720.0]              %the boundaries defining target area
  /StructParent 1
  /Type/Annot                                          %Structure element is an annotation
  /Subtype/Link
 >>                                                    %It is a link annotation
endobj
31 0 obj                                               %Object 31, gen 0, obj
 <</S/URI                                              %Object type is URI action
   /URI(http://www.w3.org/WAI)                         %The Uniform resource identifier to r
 >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8.4.4.2 (Link Elements) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Related Techniques

- G53: Identifying the purpose of a link using link text combined with the text of the enclosing sentence
- G91: Providing link text that describes the purpose of a link
- PDF13: Providing replacement text using the /Alt entry for links in PDF documents

Tests

Procedure

For each hyperlink, verify that the link is correctly tagged and the link text is properly exposed:

1. Read the PDF document with a screen reader, listening to hear that the link is read correctly and that it describes the purpose of the link (i.e., its destination).
2. Visually scan the tag tree to verify that the link is tagged correctly and the link text is exposed (for screen magnifier users and sighted users with cognitive disabilities).

3. Use a tool that is capable of showing the /Link entry value to open the PDF document and view the hyperlink and link text.
4. Use a tool that exposes the document through the accessibility API and verify that the link has the correct link text.
5. Tab to each link and check that it can be followed to its target by pressing Enter.

Expected Results

- #1 or #2 or #3 or #4 is true.
- #5 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF12: Providing name, role, value information for form fields in PDF documents

Applicability

Tagged PDF documents with interactive form fields.

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
  - How to Meet 1.3.1 (Info and Relationships)
  - Understanding Success Criterion 1.3.1 (Info and Relationships)
- Success Criterion 4.1.2 (Name, Role, Value)
  - How to Meet 4.1.2 (Name, Role, Value)
  - Understanding Success Criterion 4.1.2 (Name, Role, Value)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF12. Also see PDF Technology Notes.

Description

The objective of this technique is to ensure that assistive technologies can gather information about and interact with form controls in PDF content.

The types of PDF form controls are: text input field, check box, radio button, combo box, list box, and button.

Providing name, role, state, and value information for all form components enables compatibility with assistive technology, such as screen readers, screen magnifiers, and speech recognition software used by people with disabilities.

The PDF specification defines how name, role, and value are set for form controls in Section 12.7.4 (Field Types) of PDF 1.7 (ISO 32000-1), as shown in the following table. The Comments column explains how Adobe Acrobat Pro displays the corresponding information.

Interactive Form Dictionary Entries	Used to Define	Comments
FT	Role	Controls that share field type also use field flags to set the appropriate role. In Adobe Acrobat the role for form controls is set automatically.
TU	Name	In Adobe Acrobat the TU entry value is provided via the Tooltip field in the form control's Properties dialog. This should not be confused with the T entry which is defined as the Name in Acrobat's form control properties dialog - the name field in the Properties dialog is not used to provide the name for a control when read by assistive technologies.
CA	Name (Pushbuttons only)	In Adobe Acrobat the CA entry value is provided via the label field in the form control's Properties dialog.
V	Value	The Value entry is set by the user interacting with the control, where a value is needed.
DV	Default Value	In Adobe Acrobat the DV entry value can be set in the form control's Properties dialog.

The following table describes how the role, name, value, and state are defined for PDF form controls created using Adobe Acrobat Pro. Adobe LiveCycle Designer provides the same controls as well as several additional ones: see Example 2 below.

PDF form element	**Role (FT entry)**	**Name (TU entry)**	**Value (V entry)**	**Configurable States**
Text field	Text `/Tx`	Tooltip	Default value (DV entry in field dictionary) can be set in the Properties dialog. Value is entered by user.	Read Only, Required, Multiline, Password
Check box	Check box `/Btn`	Tooltip	V entry is set to 'Yes' or 'No' depending on Checked state.	Read Only, Required, Checked
Radio button	Radio button `/Btn` (Field Flag set to 'Radio')	Tooltip	V entry is set to 'Yes' or 'No' depending on Checked state.	Read Only, Required, Checked
Combo box	Combo box `/Ch` (Field Flag set to 'Combo')	Tooltip	Default value (/DV) can be set in the Properties dialog. Value is determined by user selection.	Read Only, Required
List box	Drop-down list `/Ch`	Tooltip	Default value (/DV) can be set in the Properties dialog. Value is determined by user selection.	Read Only, Required
Button	Push button `/Btn` (Field Flag set to 'Pushbutton')	Label (CA entry instead of TU entry)	Push buttons do not have or require a value.	Read Only, Required
Signature field	Text `/Sig`	Tooltip	Default value (DV entry in field dictionary) can be set in the Properties dialog. Value is entered by user.	Read Only, Required

Examples

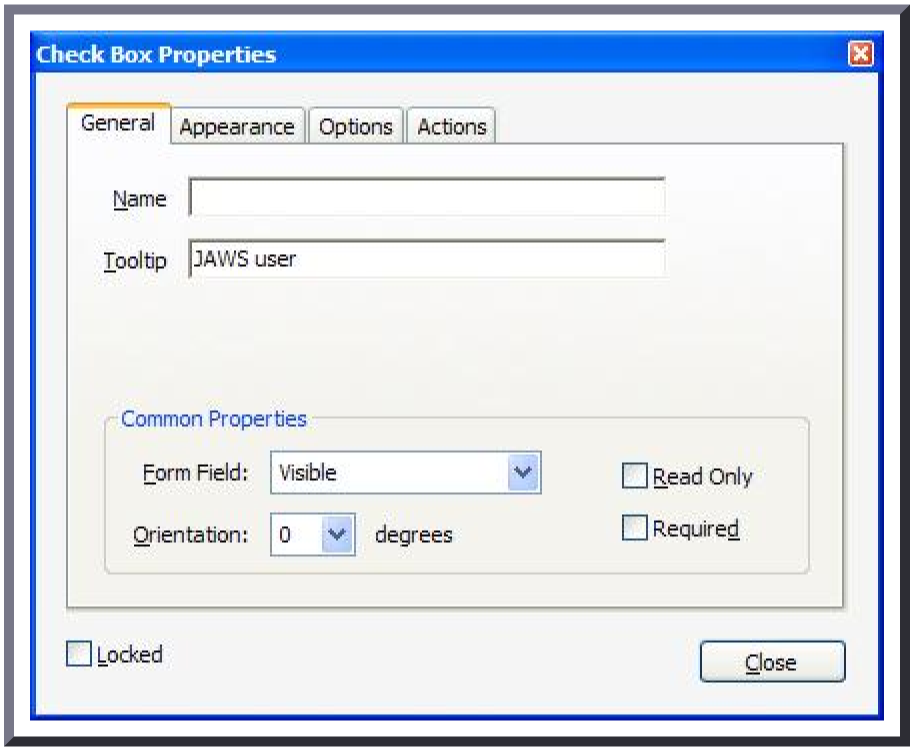
Example 1: Specifying name, role, value and/or state for a form field using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

This example uses a check box for illustration; the procedure is the same for other form controls. In Form Editing mode:

1. Access the context menu for the form field you are creating or modifying.
2. Select the Properties... dialog for the form field.
3. Specify the name by adding a value to the tool tip field. This will used by the accessibility API as the Name for the control and should usually be set to match the text used as a visual label for the control.
4. Select the Options tab.
5. Specify the default value and the default state, if appropriate.

The image below shows the Check Box Properties dialog, open in the General tab. (The Name field in the dialog is not needed for accessibility.)



Name

Tooltip JAWS user

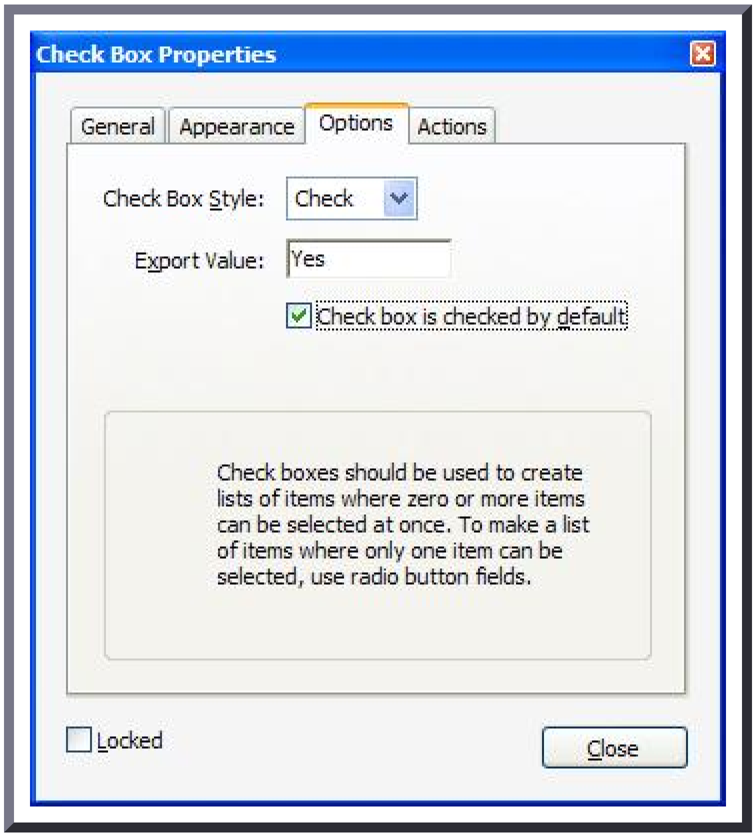
Form Field: Visible

Orientation: 0 degrees

Read Only

Required

The image below shows the Check Box Properties dialog, open in the Options tab.



Check Box Style: Check

Export Value: Yes

✓ Check box is checked by default

Check boxes should be used to create lists of items where zero or more items can be selected at once. To make a list of items where only one item can be selected, use radio button fields.

☐ Locked

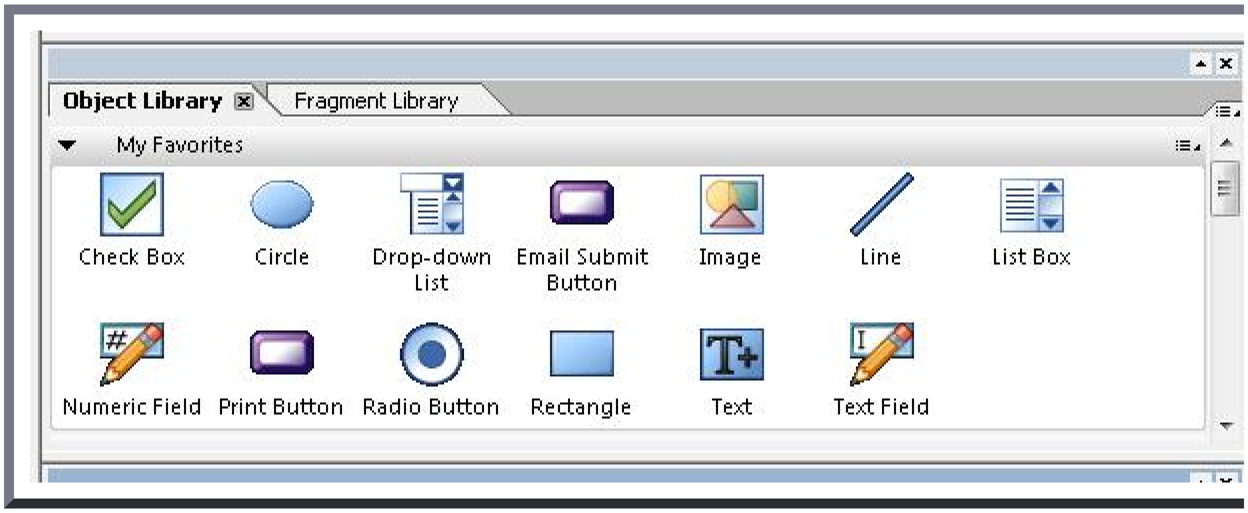
This example is shown in operation in the [working example of specifying name, role, value using Acrobat Pro](https://www.w3.org/TR/WCAG20-TECHS/pdf).

Example 2: Specifying name, value, and state for a form field using Adobe LiveCycle Designer ES 8.2.1

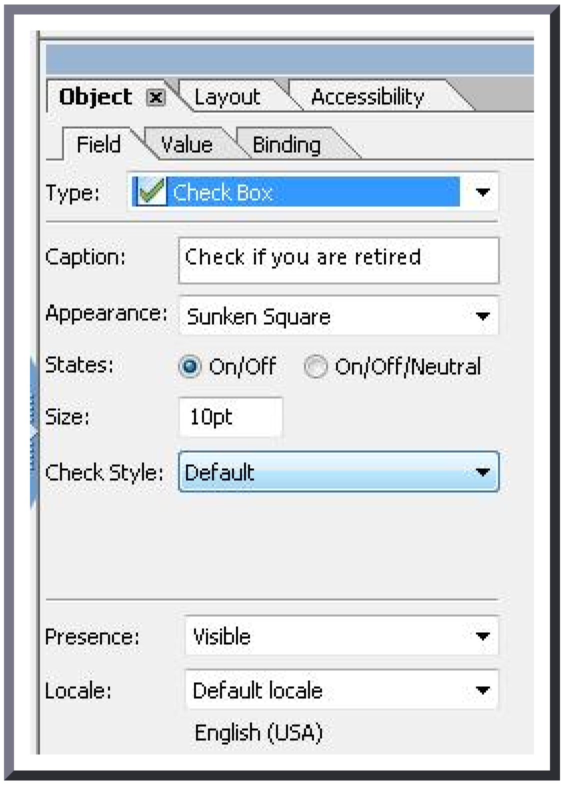
This example is shown with Adobe LiveCycle Designer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

In Adobe LiveCycle Designer, you use the Object Library to create form objects and the Object Palette to specify name, role, state or value for the object.

The following image shows the Object Palette.



The following three images show the tabs in the Object palette. In the first the Field tab is open for specifying the type (or role) of the field.



Type: Check Box

Caption: Check if you are retired

Appearance: Sunken Square

States: [✓ On/Off] On/Off/Neutral

Size: 10pt

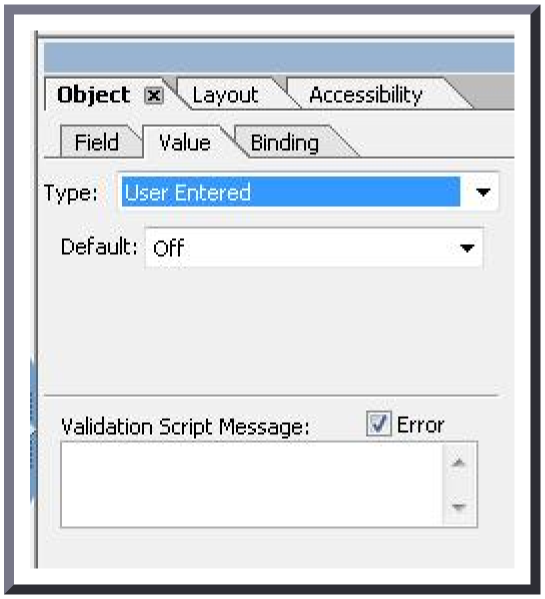
Check Style: Default

Presence: Visible

Locale: Default locale

English (USA)

The next image shows the Value tab, with options that can be applied to the field.

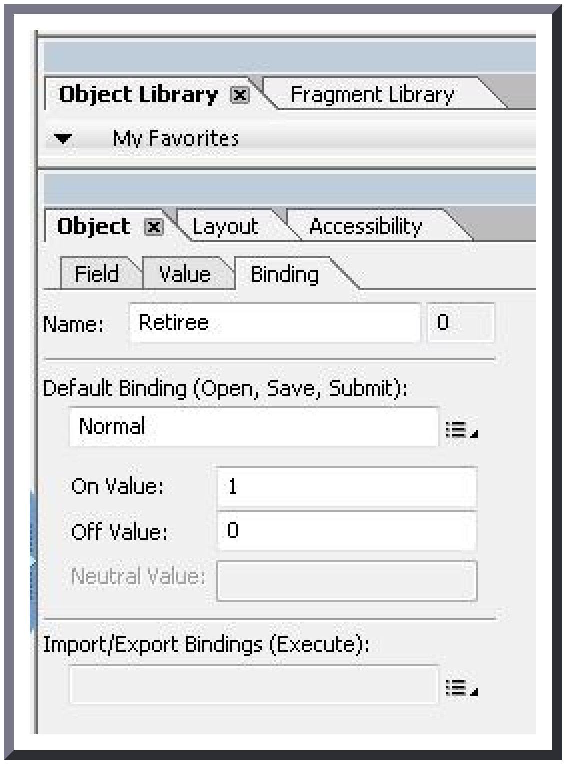


Type: User Entered

Default: Off

Validation Script Message: ✓ Error

The third images shows the Binding tab, specifying the name of the field.



Name: Retiree 0

Default Binding (Open, Save, Submit): Normal

On Value: 1

Off Value: 0

Neutral Value:

Import/Export Bindings (Execute):

This example is shown in operation in the working example of specifying name, role, value

Example 3: Adding a checkbox in a PDF document using the /Btn field type

The following code fragment illustrates code that is typical for a simple check box field such as shown in Examples 1 and 2. This is typically accomplished by an authoring tool.

```
1 0 obj
  << /FT /Btn     % Role
     /TU Retiree  % Name
     /V /Yes      % Value
     /AS /Yes
     /AP << /N << /Yes 2 0 R /Off 3 0 R>>
  >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 12.7.4 (Field Types) of PDF 1.7 (ISO 32000-1)
- Adobe XML Forms Architecture (XFA)
- PDF and Accessibility

Related Techniques

- PDF23: Providing interactive form controls in PDF documents
- PDF5: Indicating required form controls in PDF forms
- PDF22: Indicating when user input falls outside the required format or values in PDF forms

Tests

Procedure

1. For the form control, verify that name, role, and value/state are specified by one of the following:
    - Use a screen reader to navigate to the form control and check that it can be activated or that its value can be changed. Verify that the name (tooltip) and role are announced.
    - Use a tool capable of showing the form field information to open the PDF document and verify that the form control has the correct name, role, value, and state (if appropriate) information.
    - Use a tool that exposes the document through the accessibility API, and verify that the form control has the correct name, role, value, and state (if appropriate) information.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF13: Providing replacement text using the /Alt entry for links in PDF documents

Applicability

Tagged PDF documents that contain links.

This technique relates to:

- Success Criterion 2.4.4 (Link Purpose (In Context))
  - How to Meet 2.4.4 (Link Purpose (In Context))
  - Understanding Success Criterion 2.4.4 (Link Purpose (In Context))

  Note: This technique must be combined with other techniques to meet SC 2.4.4. See Understanding SC 2.4.4 for details.
- Success Criterion 2.4.9 (Link Purpose (Link Only))
  - How to Meet 2.4.9 (Link Purpose (Link Only))
  - Understanding Success Criterion 2.4.9 (Link Purpose (Link Only))

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF13. Also see PDF Technology Notes.

Description

The objective of this technique is to provide replacement link text via the /Alt entry in the property list for a tag. This is usually not necessary, but in some situations, additional information beyond the visible link text is needed, particularly for screen reader users. Screen readers can read visible link text, but replacing the screen text with meaningful alternate text for links in a PDF document can make links more accessible.

Links in PDF documents are represented by a Link tag and objects in its sub-tree, consisting of a link object reference (or Link annotation) and one or more text objects. The text object or objects inside the Link tag are used by assistive technologies to provide a name for the link.

Authors can replace the default link text by providing an /Alt entry for the Link tag. When the Link tag has an /Alt entry, screen readers ignore the value of any visible text objects in the Link tag and use the /Alt entry value for the link text.

The simplest way to provide context-independent link text that complies with the WCAG 2.0 success criteria is to create them when authoring the document, before conversion to PDF. In some cases, it may not be possible to create the links using the original authoring tool. When editing PDF documents with Adobe Acrobat Pro, the best way to create accessible links is to use the Create Link command.

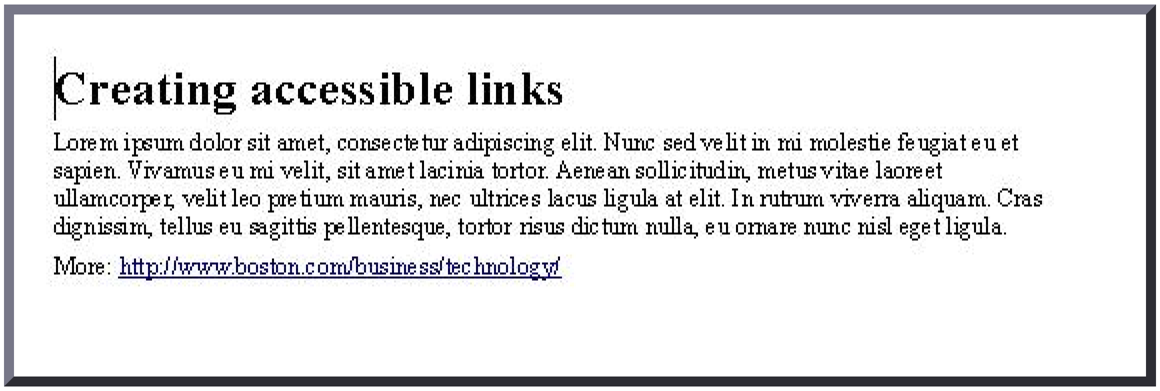
Authors should make sure that the alternate text makes sense in context of the screen text before and after the link.

Examples

Example 1: Adding alternate link text using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

The image below shows a document converted to PDF from Oracle Open Office. Note that the visible link text is the URL for the link target. A screen reader will read the entire URI as the link text.



Creating accessible links

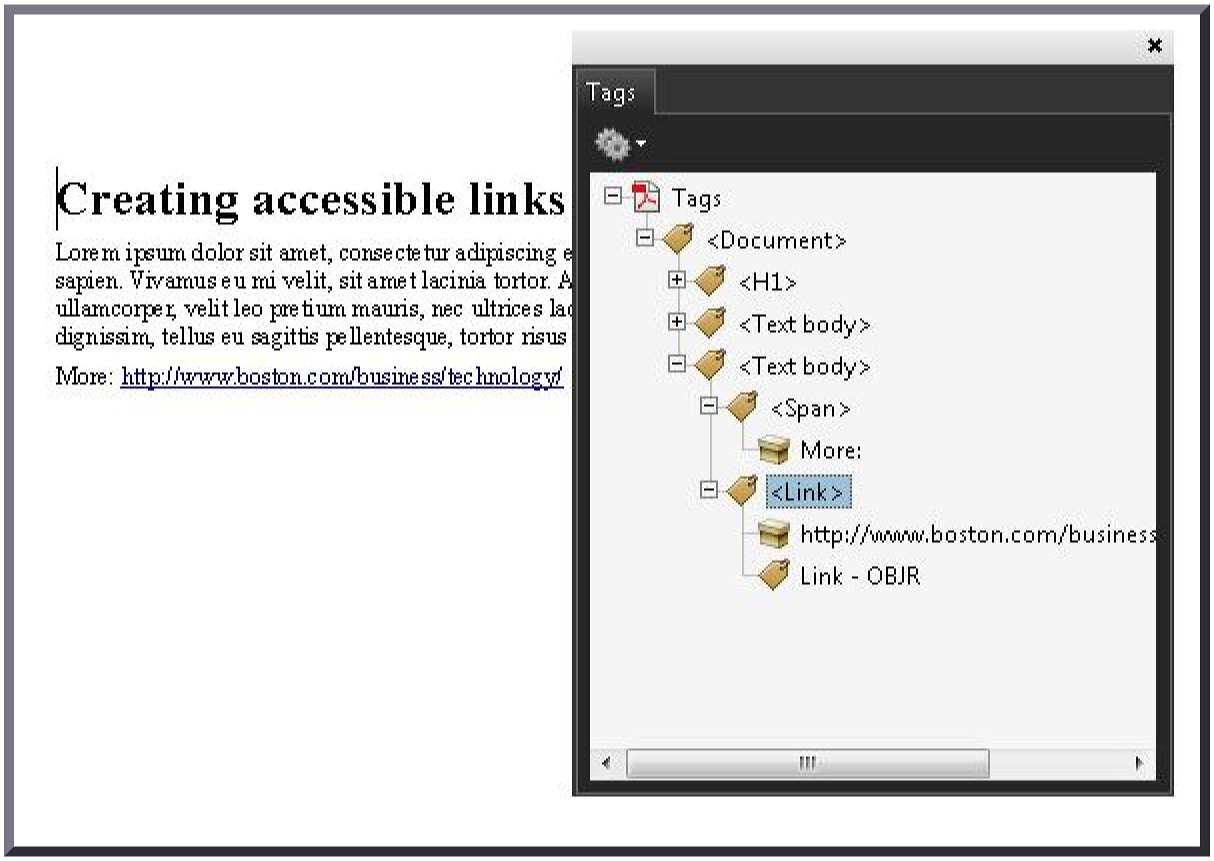
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc sed velit in mi molestie feugiat eu et sapien. Vivamus eu mi velit, sit amet lacinia tortor. Aenean sollicitudin, metus vitae laoreet ullamcorper, velit leo pretium mauris, nec ultrices lacus ligula at elit. In rutrum viverra aliquam. Cras dignissim, tellus eu sagittis pellentesque, tortor risus dictum nulla, eu ornare nunc nisl eget ligula.

More: http://www.boston.com/business/technology/

To create more accessible link text for assistive technology:

1. In the View menu, open the Tag panel by selecting Navigation Panels > Tags.
2. Locate the Link tag in the tag tree, access the context menu for the link, and select Properties.
3. In the TouchUp Properties dialog, in the Tags tab, enter replacement text in the Alternate Text field. Screen readers will read this text instead of the entire URI.

The next image shows the Link tag structure in the Tag panel.



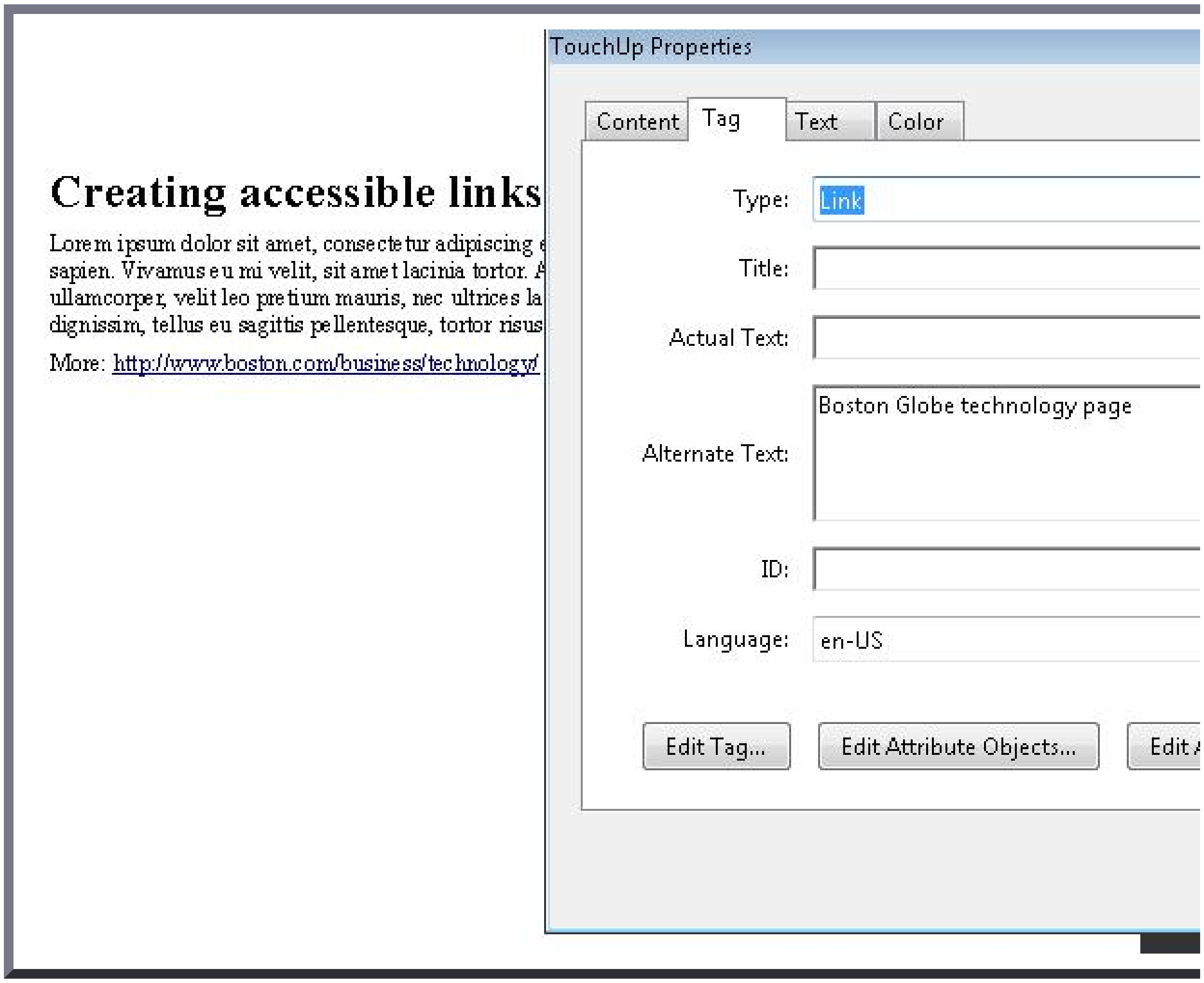
Creating accessible links

Lorem ipsum dolor sit amet, consectetur adipiscing e
sapien. Vivamus eu mi velit, sit amet lacinia tortor. A
ullamcorper, velit leo pretium mauris, nec ultrices la
dignissim, tellus eu sagittis pellentesque, tortor risus

More: http://www.boston.com/business/technology/

```
Tags
  <Document>
    <H1>
    <Text body>
    <Text body>
      <Span>
        More:
      <Link>
        http://www.boston.com/business
        Link - OBJR
```

The last image shows the Alternate Text specified in the Link tag's TouchUp Properties dialog. A screen reader will read the Alternate Text as the link text.



Creating accessible links

Lorem ipsum dolor sit amet, consectetur adipiscing e
sapien. Vivamus eu mi velit, sit amet lacinia tortor. A
ullamcorper, velit leo pretium mauris, nec ultrices la
dignissim, tellus eu sagittis pellentesque, tortor risus

More: http://www.boston.com/business/technology/

TouchUp Properties

Type: Link

Title:

Actual Text:

Alternate Text: Boston Globe technology page

ID:

Language: en-US

This example is shown in operation in the working example of adding alternate link text (OpenOffice file) and working example of adding alternate link text (PDF file).

Example 2: Adding alternate link text in a PDF document using the /Alt entry

The following code fragment illustrates code that is typical for alternative text for a link. This is typically accomplished by an authoring tool.

```
32 0 obj
<<
  /S/URI                                       %Action type (required), must be URI for a UR
  /URI(http://www.boston.com/business/technology/)  %Uniform resource identifier(required),
>>
endobj
```

The following illustrates how to specify alternate text for the URL in the above link:

```
11 0 obj
<<
  /Alt(Boston Globe technology page)    %Alternate text entry
  /K [ 1
       <<
         /Obj 27 0 R
         /Type /OBJR            %Object reference to the link
       >>
       ]
  /P 12 0 R
  /Pg 18 0 R
  /S
  /Link
>>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.9.4 (Replacement Text) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Related Techniques

- G53: Identifying the purpose of a link using link text combined with the text of the enclosing sentence
- G91: Providing link text that describes the purpose of a link
- G149: Using user interface components that are highlighted by the user agent when they receive focus
- PDF11: Providing links and link text using the Link annotation and the /Link structure element in PDF documents

Tests

Procedure

1. For the hyperlink, verify that the alternate link text is properly coded by one of the following:
   - Read the PDF document with a screen reader, listening to hear that the alternate link text is read correctly.
   - Use a tool that is capable of showing the /Alt entry to open the PDF document and view the hyperlink and alternate link text.
   - Use a tool that exposes the document through the accessibility API and verify that the alternate link text is the text for the link.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF14: Providing running headers and footers in PDF documents

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 2.4.8 (Location)
  - How to Meet 2.4.8 (Location)
  - Understanding Success Criterion 2.4.8 (Location)
- Success Criterion 3.2.3 (Consistent Navigation)
  - How to Meet 3.2.3 (Consistent Navigation)
  - Understanding Success Criterion 3.2.3 (Consistent Navigation)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF14. Also see PDF Technology Notes.

Description

The objective of this technique is to help users locate themselves in a document by providing running headers and footers via pagination artifacts. This is normally accomplished using a tool for authoring PDF.

Running headers and footers help make content easier to use and understandable by providing repeated information in a consistent and predictable way. The content of headers and footers will vary widely depending on the document scope and content, the audience, and design decisions. Some examples of location information that may be used in headers and footers are listed below. Whether the information appears in a header or a footer is often a design decision; page numbers often appear in footers but they may alternatively appear in headers.

- Document title
- Current chapter and/or section in the document
- Page numbers with location information such as, "Page 3-4" or "Page 9 of 15."
- Author and/or date information.

Consistency helps users with cognitive limitations, screen-reader users and low-vision magnifier users, and users with intellectual disabilities understand content more readily.

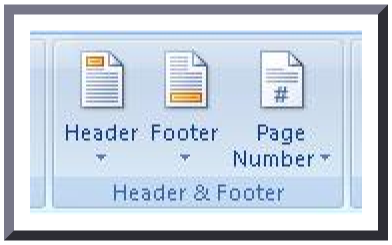
The easiest way to provide page headers and footers is in the authoring tool for the document. Authoring tools typically provide features for creating header and footer text and information (such as page numbers). However, if after converting your document to PDF, you need to add or modify page headers and footers, authoring or repair tools like Adobe Acrobat Pro's Header & Footer tools can be used. In all cases, the tools generate page headers and footers in consistent and predictable layout, format, and text.

Examples

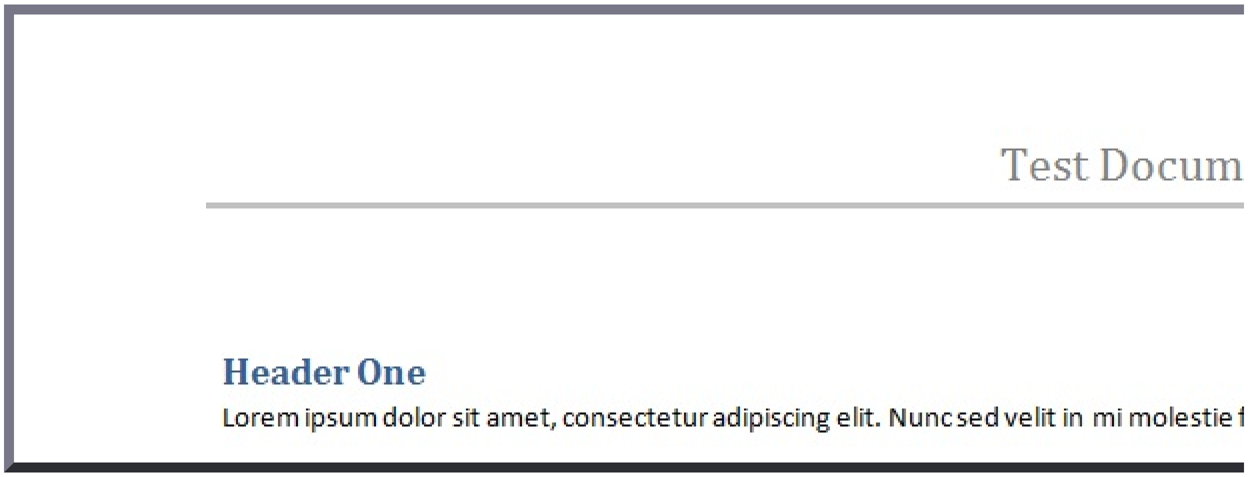
Example 1: Adding running headers and footers using Microsoft Word 2007

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

In Microsoft Word, use the Insert ribbon, which allows you to specify header, footer, and page number information and layout, as shown in the following images.



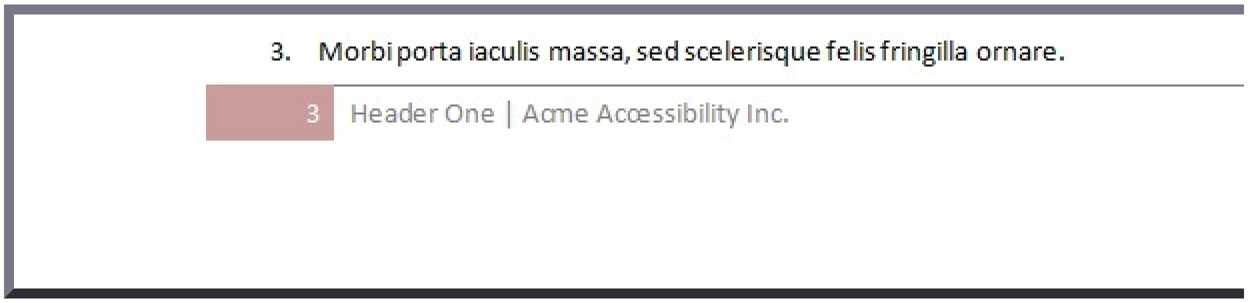
You can use these tools to specify headers and footers as shown in the following images:



Test Docum

Header One

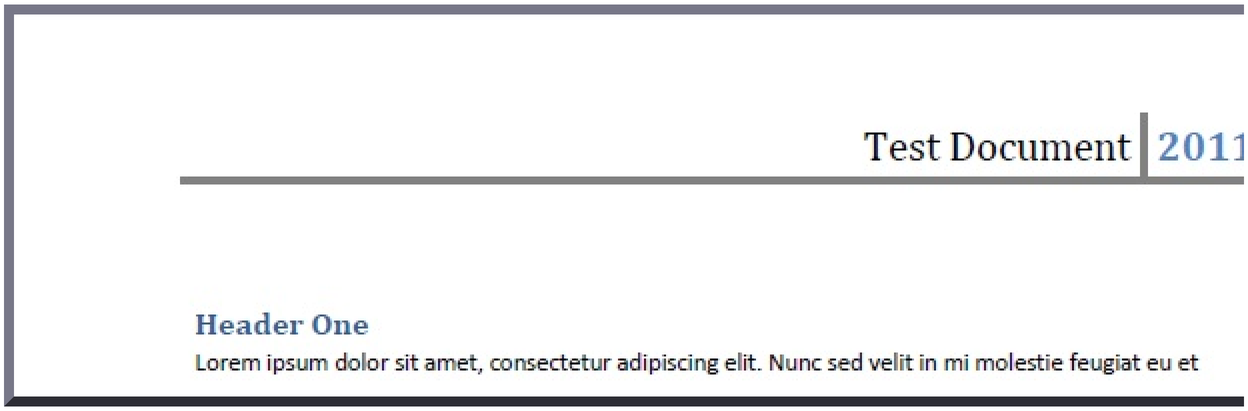
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc sed velit in mi molestie f



3. Morbi porta iaculis massa, sed scelerisque felis fringilla ornare.

3 Header One | Acme Accessibility Inc.

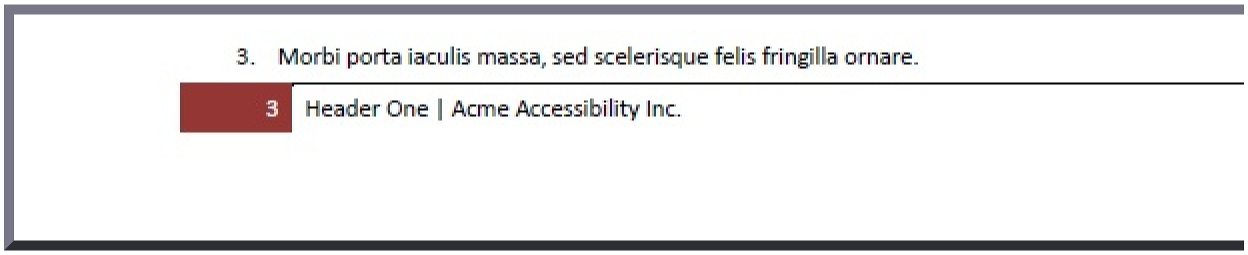
When converted to PDF, the page headers and footers appear in the document.



Test Document 2011

Header One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc sed velit in mi molestie feugiat eu et



3. Morbi porta iaculis massa, sed scelerisque felis fringilla ornare.

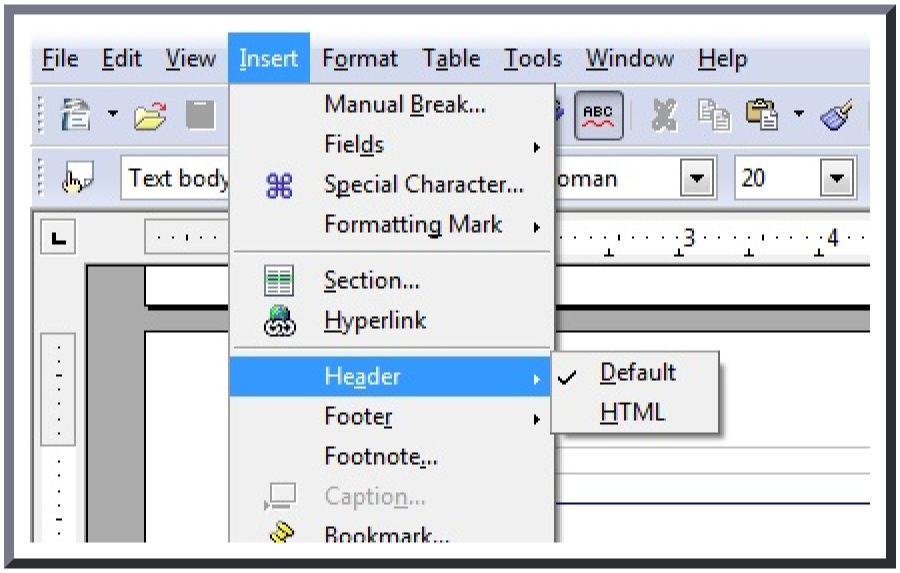
3 Header One | Acme Accessibility Inc.

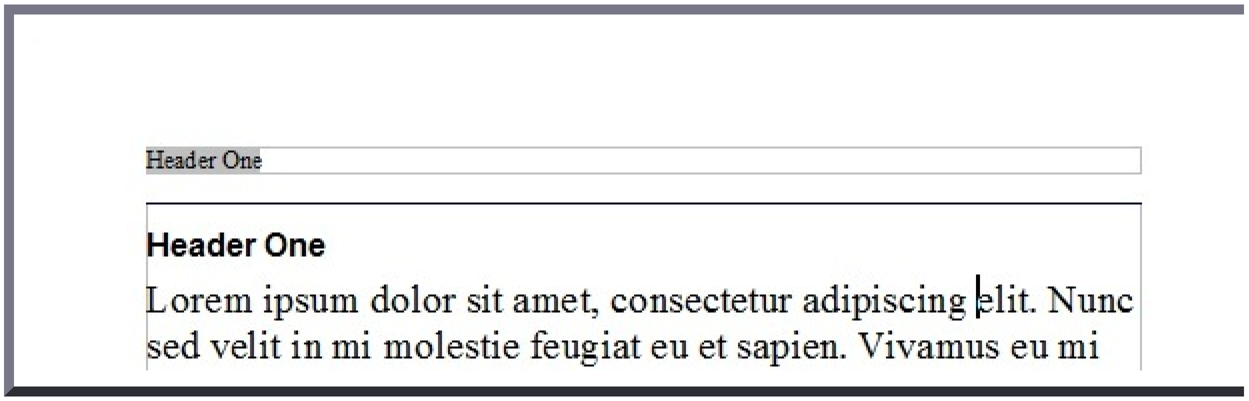
This example is shown in operation in the working example of adding running headers using Word (Word file) and working example of adding running headers using Word (PDF file).

Example 2: Adding running headers and footers using OpenOffice.org Writer 2.2

This example is shown with OpenOffice.org Writer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

In OpenOffice.org Writer, use the Insert > Header and Insert > Footer tools, which allow you to specify header and footer information and layout, as shown in the following images.

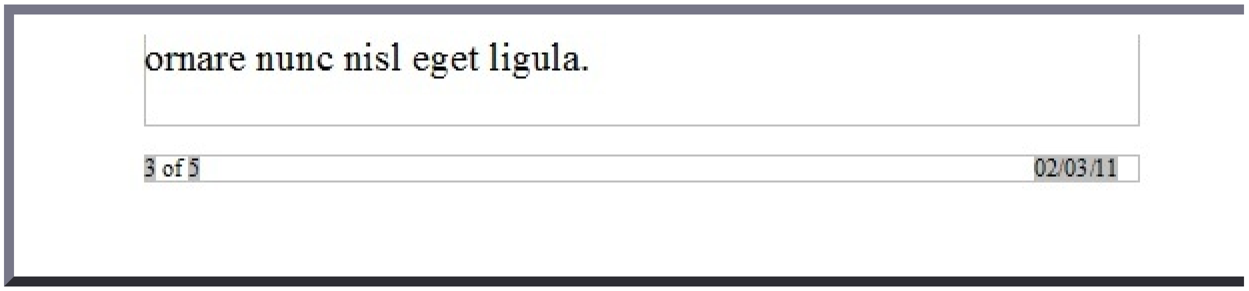




Header One

Header One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc sed velit in mi molestie feugiat eu et sapien. Vivamus eu mi



ornare nunc nisl eget ligula.

3 of 5 02/03/11

When converted to PDF, the page headers and footers appear in the document as they do in the converted Word document in Example 1.

This example is shown in operation in the working example of adding running headers using OpenOffice Writer (OpenOffice file) and working example of adding running headers using

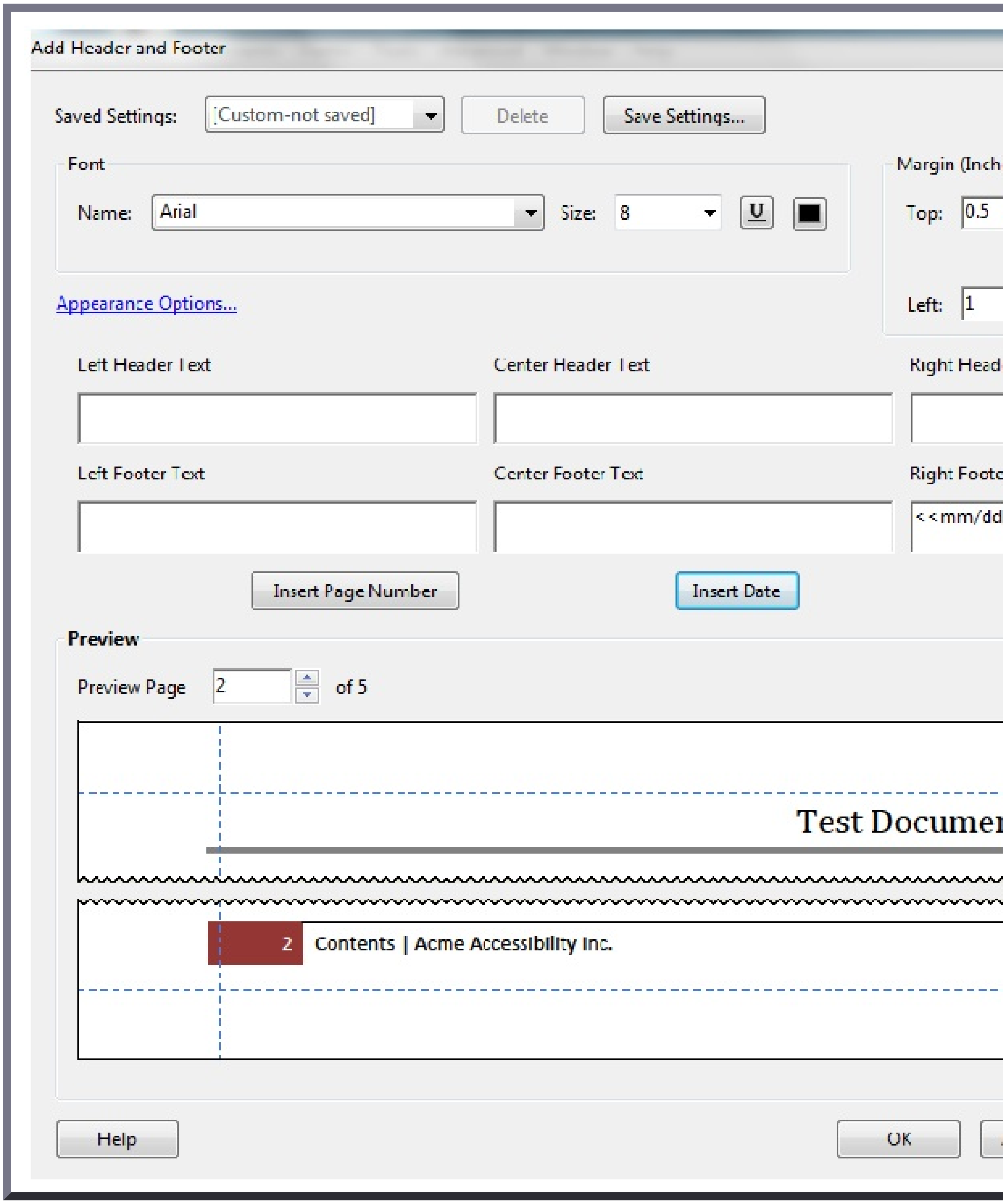
Example 3: Adding running headers and footers to PDF documents using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

In Adobe Acrobat Pro, you can add or modify headers and footers:

1. Select Document > Header & Footer > Add...
2. In the Add Header and Footer tool, specify text and formats for headers and footers in your document.
3. Use the Previews to make sure the text, fonts, and layout are as you want them for your document.

The image below shows Acrobat Pro's Add Header and Footer tool.



Add Header and Footer

Saved Settings: [Custom-not saved]

Font

Name: Arial

Size: 8

Margin (Inch

Top: 0.5

Left: 1

Left Header Text

Center Header Text

Right Head

Left Footer Text

Center Footer Text

Right Foote

<<mm/dd

Preview

Preview Page 2 of 5

Test Documen

2 Contents | Acme Accessibility Inc.

Example 4: Marking a running header or footer as a pagination artifact in a PDF document using an /Artifact tag or property list

The PDF specification allows running headers and footers to be marked as "pagination artifacts" as defined in section 14.8.2.2 "Real Content and Artifacts," of PDF 1.7 (ISO 32000-1).

An artifact is explicitly distinguished from real content by enclosing it in a marked-content sequence with the /Artifact tag.

```
/Artifact
BMC
...
EMC
```

or

```
/Artifact propertyList
BDC
...
EMC
```

The first is used to identify a generic artifact; the second is used for artifacts that have an associated property list. Note: to aid in text reflow, artifacts should be defined with property lists whenever possible. Artifacts lacking a specified bounding box are likely to be discarded during reflow.

Property list entries for artifacts include Type, BBox, Attached, and Subtype.

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8.2.2 (Real Content and Artifacts) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Related Techniques

- G61: Presenting repeated components in the same relative order each time they appear
- PDF9: Providing headings by marking content with heading tags in PDF documents
- PDF2: Creating bookmarks in PDF documents

Tests

Procedure

1. Check that running headers and/or footers are provided and contain information to help users locate themselves within the document (such as page numbers or chapter numbers).
2. If section headers are used in the running header or footer, check that the section header and the running header or footer are consistent.

Expected Results

- #1 and #2 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF15: Providing submit buttons with the submit-form action in PDF forms

Applicability

Tagged PDF documents with forms.

This technique relates to:

- Success Criterion 3.2.2 (On Input)

- How to Meet 3.2.2 (On Input)
- Understanding Success Criterion 3.2.2 (On Input)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF15. Also see PDF Technology Notes.

Description

The objective of this technique is to provide a mechanism that allows users to explicitly request a change of context using the submit-form action in a PDF form. The intended use of a submit button is to generate an HTTP request that submits data entered in a form, so it is an appropriate control to use for causing a change of context. In PDF documents, submit buttons are normally implemented using a tool for authoring PDF.

Examples 1 and 2 demonstrate how to add a submit button using specific authoring tools. There are other PDF tools that perform similar functions. Check the functionality provided by PDF Authoring Tools that Provide Accessibility Support.

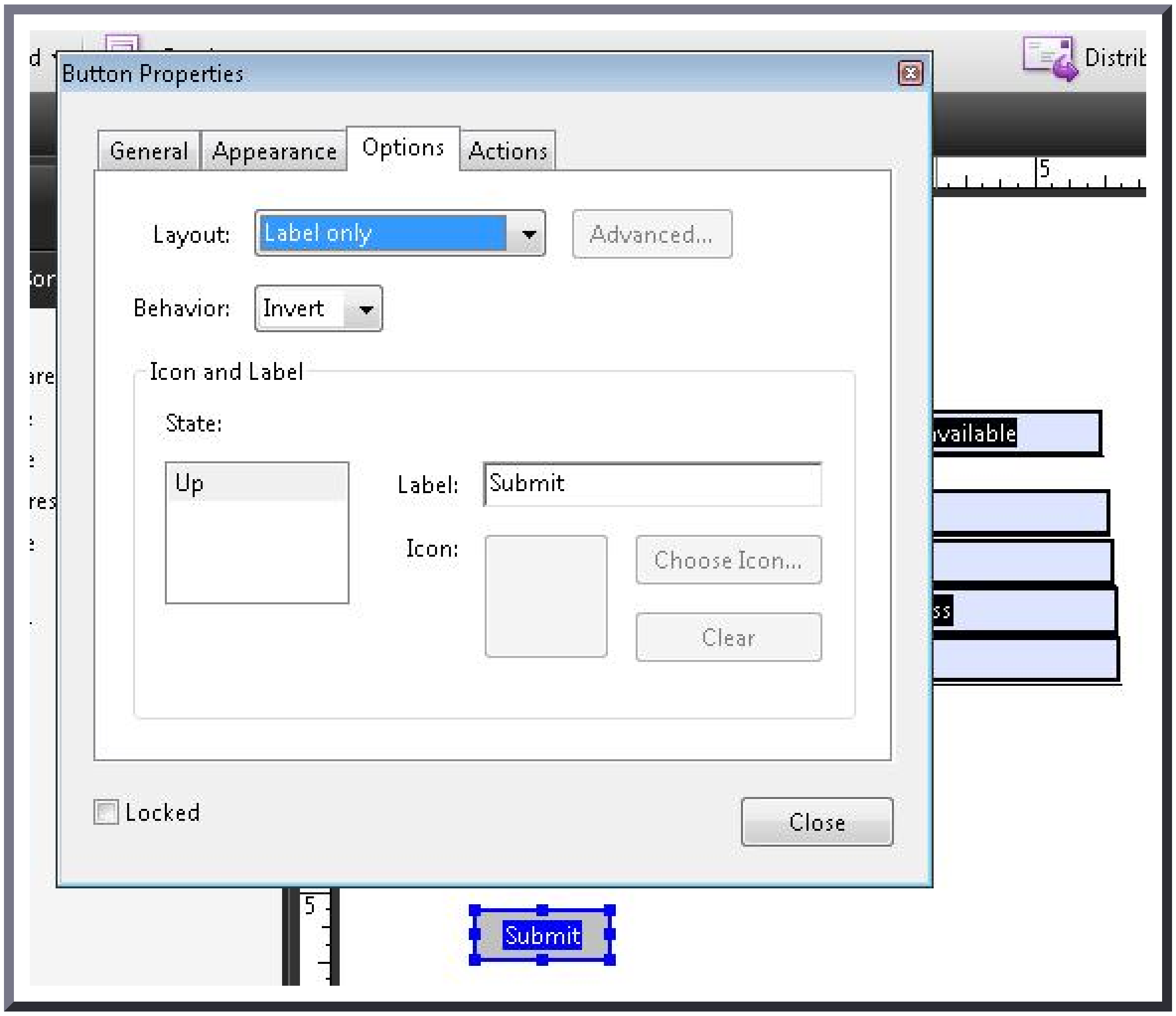
Examples

Example 1: Adding a submit button using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. From the toolbar, select Forms > Form Tools > Button and create a button on the form.
2. Access the context menu for the button and select Properties... to open the Button Properties dialog.
3. In the General tab, provide a tooltip for the button.
4. In the Options tab, choose an option in the Layout menu for the button label, icon image, or both. Then, type text in the Label box to identify the button as a submit button and/or click Choose Icon and locate the image file you want to use.
5. In the Actions tab:
   - For Select Trigger, choose Mouse Up. (The Mouse Up event is keyboard accessible and, in addition, ensures that the button will not change context unexpectedly, as it might with, e.g., a Mouse Enter event.)
   - For Select Action, choose Submit A Form.
   - Click Add.
6. In the Add dialog, enter a URL to collect data on a server or collect form data as e-mail attachments.

The following image shows the Options tab on the Button Properties dialog.


Button Properties

General | Appearance | Options | Actions

Layout: Label only

Advanced...

Behavior: Invert

Icon and Label

State:

Up

Label: Submit

Icon:

Choose Icon...

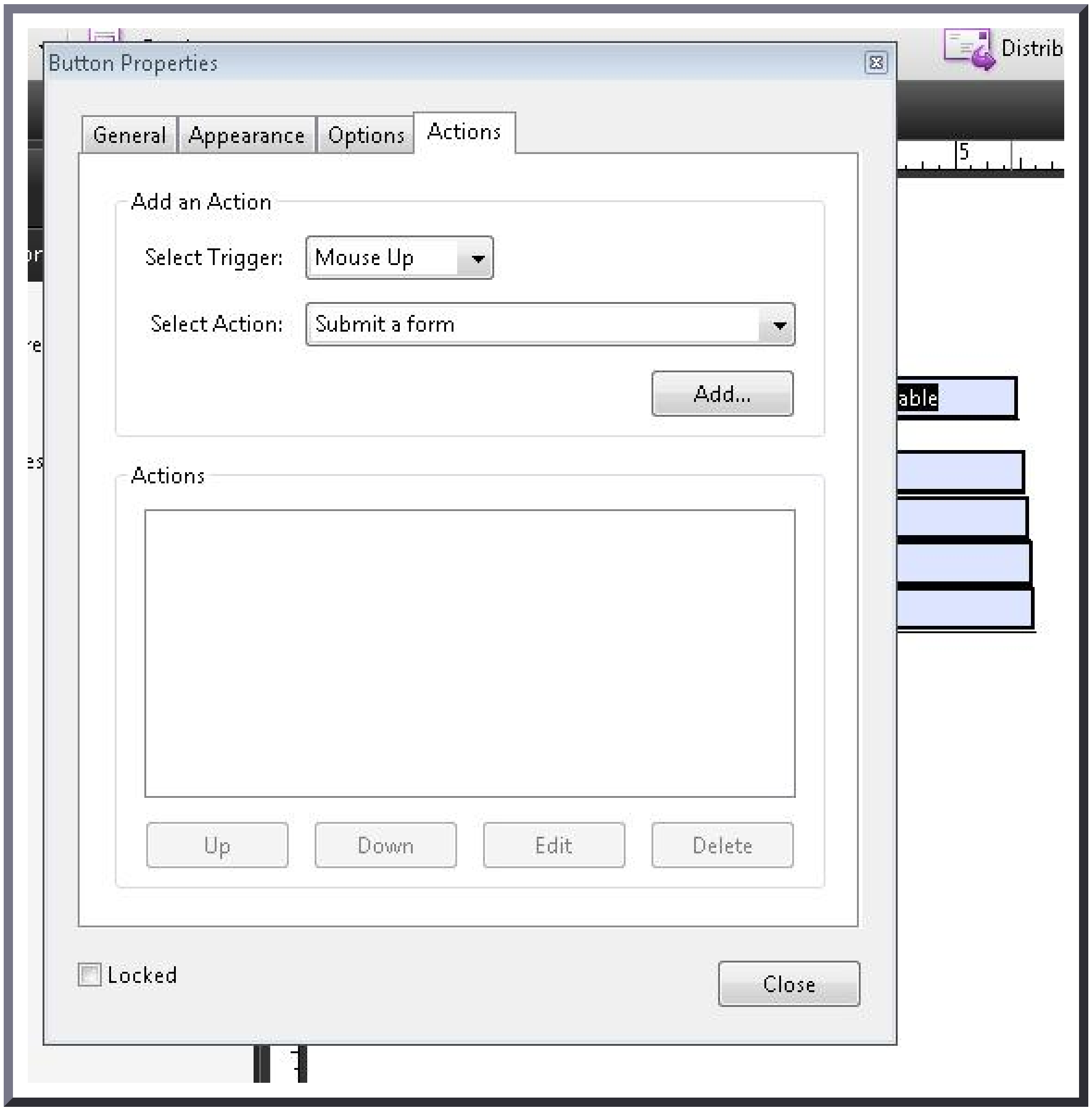
Clear

Locked

Close

Submit

The following image shows the Actions tab on the Button Properties dialog.



Select Trigger: Mouse Up

Select Action: Submit a form

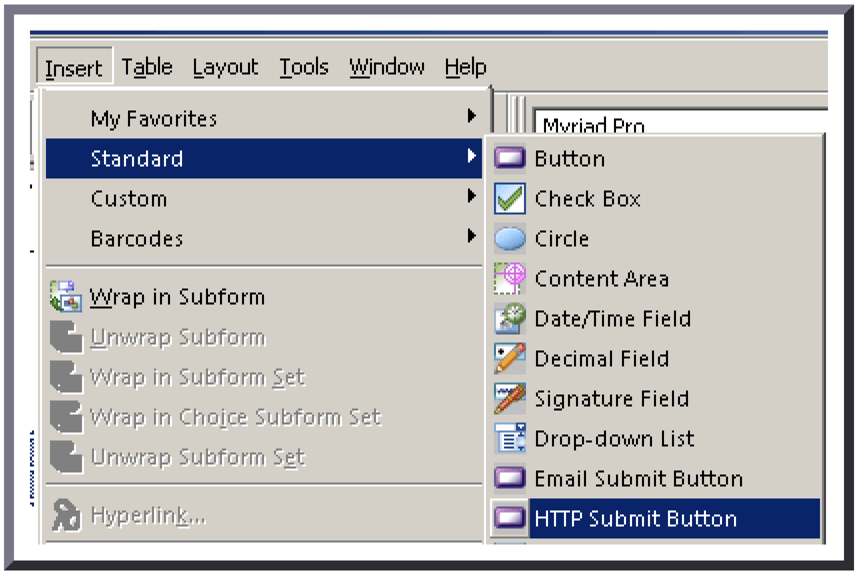
Locked

Example 2: Adding a submit button using Adobe LiveCycle Designer ES 8.2.1

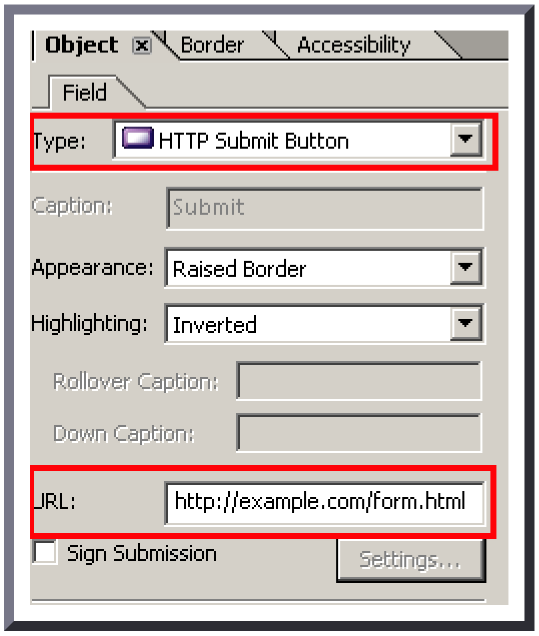
This example is shown with Adobe LiveCycle Designer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. On the Insert > Standard menu, select the HTTP Submit Button item.
2. On the Object panel for the HTTP Submit Button, insert the URL for form-submission processing.

The following image shows the Standard menu with the list of form controls.



The following image shows the Object panel with the URL and other fields for button appearance.



Type: HTTP Submit Button

Caption: Submit

Appearance: Raised Border

Highlighting: Inverted

Rollover Caption:

Down Caption:

URL: http://example.com/form.html

Sign Submission

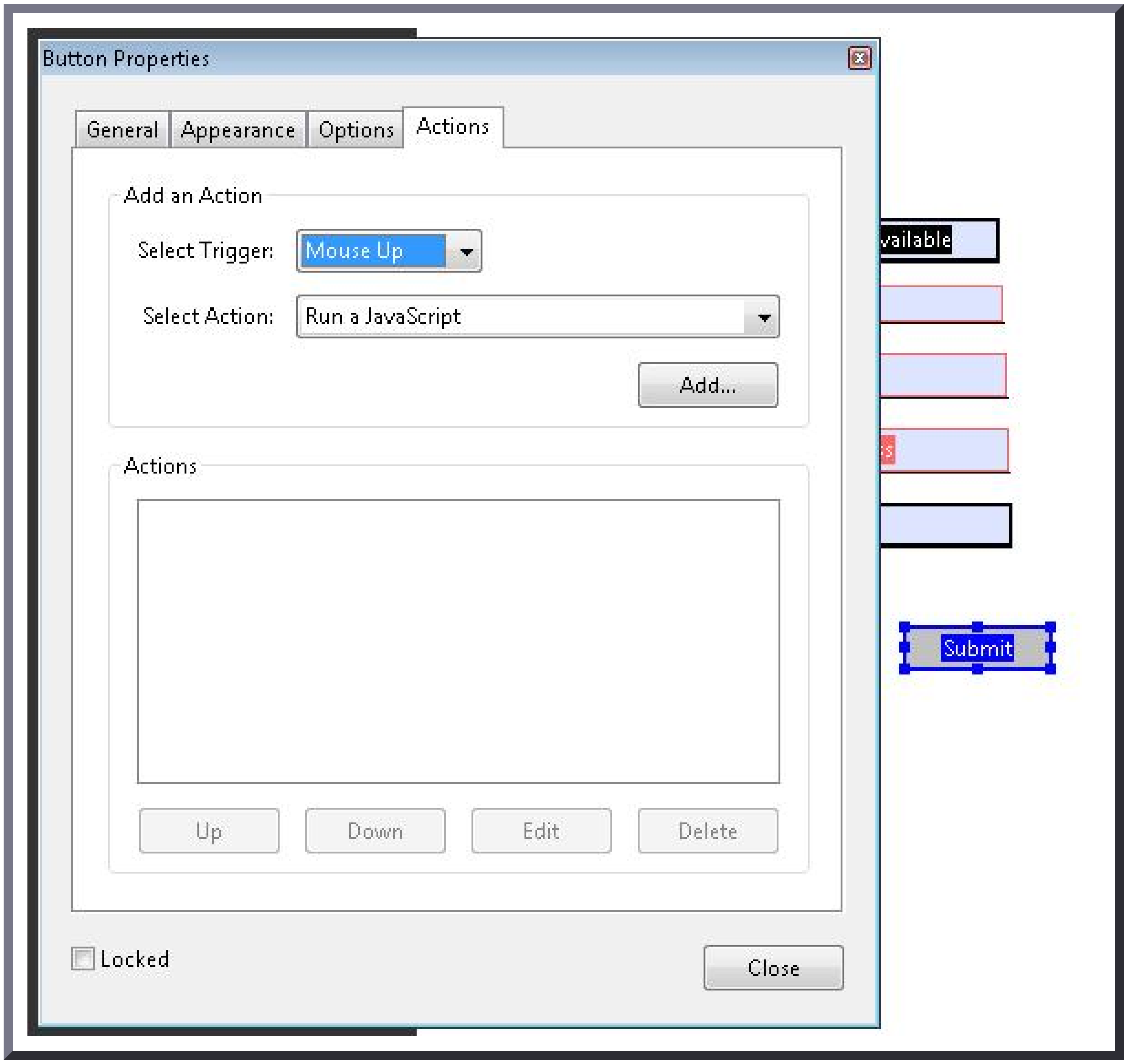
Example 3: Adding a script action to a submit button in a PDF document using JavaScript

The following JavaScript code illustrates the use of a script to specify the submit-form action. To add this script to the form field:

1. Open the Button Properties dialog, as shown in Example 1, and select the Actions tab
2. Select Run a JavaScript from the drop-down list, and select the Add button
3. Enter JavaScript code in the JavaScript Editor dialog, for example:

```
var aSubmitFields = new Array( "name", "id", "juser" );
this.submitForm({
  cURL: "http://www.example.com/cgi-bin/myscript.cgi#FDF",
  aFields: aSubmitFields,
  cSubmitAs: "FDF" // the default, not needed here
});
```

The following images illustrate this process:



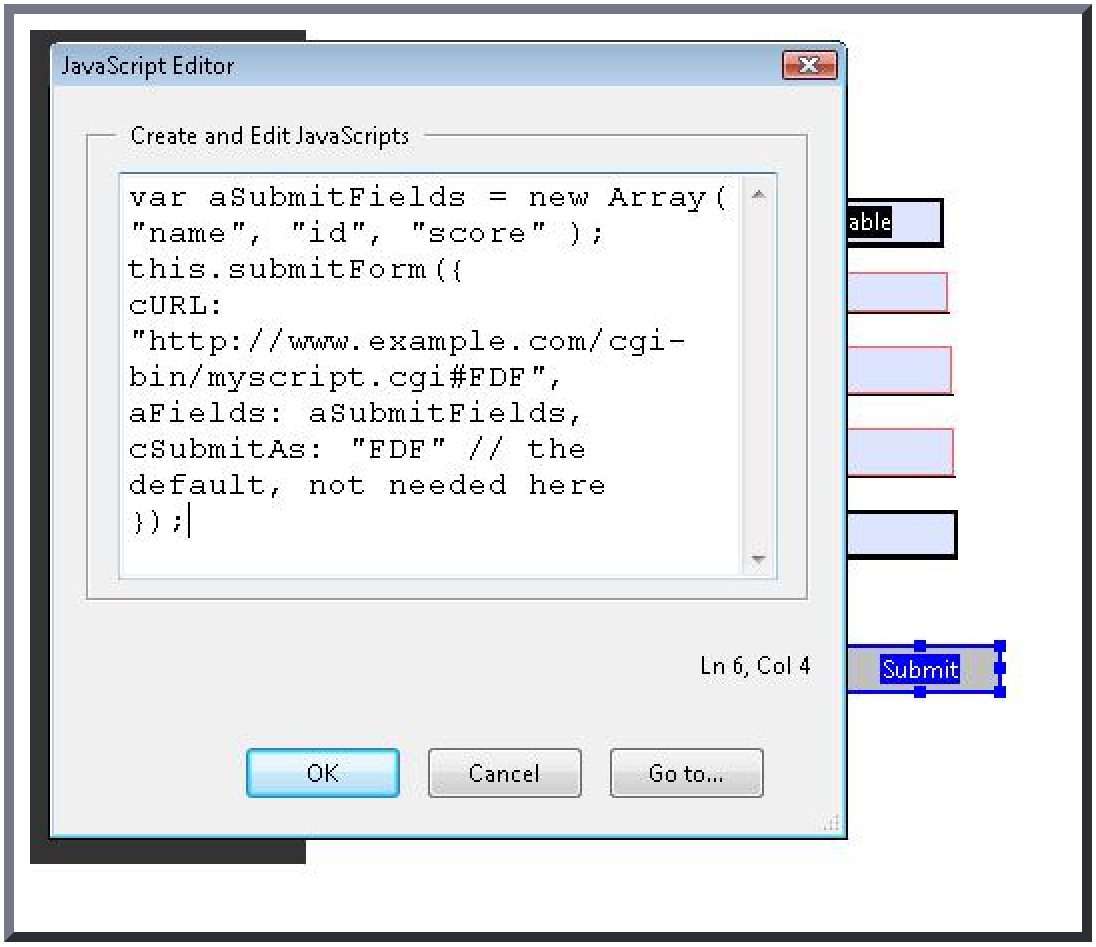
Add an Action

Select Trigger: Mouse Up

Select Action: Run a JavaScript

Actions

[ ] Locked



```
var aSubmitFields = new Array(
"name", "id", "score" );
this.submitForm({
cURL:
"http://www.example.com/cgi-
bin/myscript.cgi#FDF",
aFields: aSubmitFields,
cSubmitAs: "FDF" // the
default, not needed here
});
```

This example is shown in operation in the working example of adding a script action to a submit button.

Resources

Resources are for information purposes only, no endorsement implied.

- Section 12.7.5.2 (Submit-Form Action) in PDF 1.7 (ISO 32000-1)
- Create submission forms in LiveCycle Designer
- XML Forms Architecture (XFA) Specification Version 2.5
- PDF and Accessibility

Related Techniques

- G80: Providing a submit button to initiate a change of context
- PDF23: Providing interactive form controls in PDF documents

- PDF12: Providing name, role, value information for form fields in PDF documents

Tests

Procedure

1. For each page that submits a form, visually verify that the form contains a submit button and check one of the following:
    - Tab to the button and check that it submits the form in response to user action to select the button.
    - Open the PDF document with a tool that is capable of showing the submit-form action and check that the button action is to submit the form.

Expected Results

- #1 is true for each page that contains a form.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF16: Setting the default language using the /Lang entry in the document catalog of a PDF document

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 3.1.1 (Language of Page)
    - How to Meet 3.1.1 (Language of Page)
    - Understanding Success Criterion 3.1.1 (Language of Page)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF16. Also see PDF Technology Notes.

Description

The objective of this technique is to specify a document's default language by setting the /Lang entry in the document catalog. This is normally accomplished using a tool for authoring PDF.

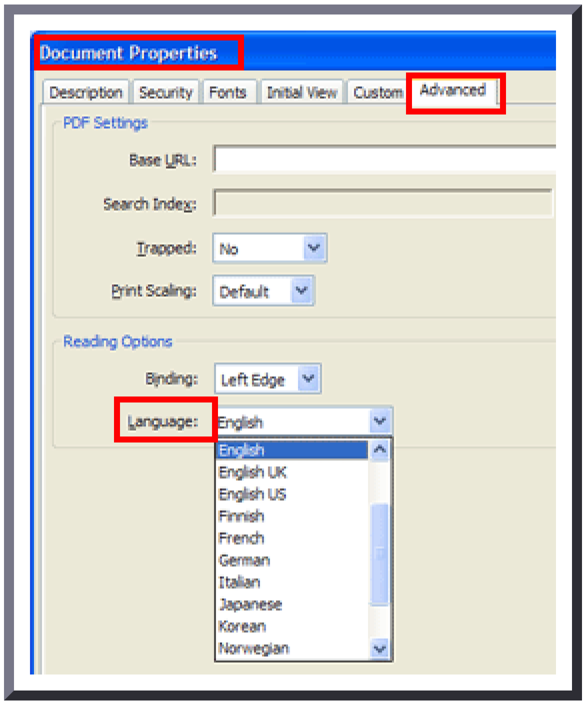
Both assistive technologies and conventional user agents can render text more accurately when the language of the document is identified. Screen readers can load the correct pronunciation rules. Visual browsers can display characters and scripts correctly. Media players can show captions correctly. As a result, users with disabilities are better able to understand the content.

Examples

Example 1: Adding a /Lang entry to specify the default document language using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Open the document in Adobe Acrobat Pro
2. From the File menu, select "Properties..."
3. In the "Properties" dialog, select the "Advanced" tab
4. In the "Reading Options" field, select the default language from the "Language" combo box



Base URL:
Search Index:
Trapped: No
Print Scaling: Default
Binding: Left Edge
Language: English

Note: Acrobat includes 16 preset language selections. If you need to specify a language that is not on the list, such as Russian, you must type the ISO 639 code for the language, not its name.

This example is shown in operation in the working example of adding a /Lang entry using Acrobat Pro.

Example 2: Specifying the default document language in a PDF document using Microsoft Word 2007

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Documents authored in Microsoft Word: "In some instances, even if the document language has been specified in the source file, the information about document language is not conveyed to the PDFMaker. Setting the language for an entire document in the Document Properties dialog box [see Example 1] corrects all errors related to this option."(Adobe® Acrobat® 9 Pro Accessibility Guide: Creating Accessible PDF from Microsoft® Word)

Example 3: Specifying the default document language in a PDF document using a /Lang entry

The natural language used for text in a document is determined in a hierarchical fashion, based on whether an optional /Lang entry is present in any of several possible locations. At the highest level, the document's default language may be specified by a /Lang entry in the document catalog.

The following code fragment illustrates code that is typical for using the /Lang entry in the document catalog for a document's default language (in this case, US English). (This is typically accomplished by an authoring tool.)

```
1 0 obj
  << /Type /Catalog
     ...
     /Lang (en-US)
     ...
  >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.9.2 (Natural Language Specification) in PDF 1.7 (ISO 32000-1)
- ISO 639-2 Codes

- PDF Reference 1.6, 10.8.1 Natural Language Specification (PDF 8.7 Mb)
- PDF Standards: Natural Language Specification
- Adobe® Acrobat® 9 Pro Accessibility Guide: Creating Accessible PDF from Microsoft® Word
- PDF and Accessibility

Related Techniques

- PDF19: Specifying the language for a passage or phrase with the Lang entry in PDF documents

Tests

Procedure

1. Verify that the default language for the document is correctly specified by applying one of the following:
   - Read the PDF document with a screen reader, listening to hear that the text is read in the correct natural language.
   - Using a PDF editor, check that the language is set to the default document language.
   - Use a tool which is capable of showing the /Lang entry value in the document catalog to open the PDF document and view the language settings.
   - Use a tool that exposes the document through the accessibility API and verify that the language is set to the default language.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF17: Specifying consistent page numbering for PDF documents

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
  - How to Meet 1.3.1 (Info and Relationships)
  - Understanding Success Criterion 1.3.1 (Info and Relationships)
- Success Criterion 2.4.8 (Location)
  - How to Meet 2.4.8 (Location)
  - Understanding Success Criterion 2.4.8 (Location)
- Success Criterion 3.2.3 (Consistent Navigation)
  - How to Meet 3.2.3 (Consistent Navigation)
  - Understanding Success Criterion 3.2.3 (Consistent Navigation)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF17. Also see PDF Technology Notes.

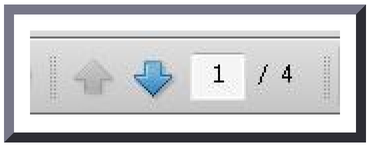
Description

The objective of this technique is to help users locate themselves in a document by ensuring that the page numbering displayed in the PDF viewer page controls has the same page numbering as the document. For example, Adobe Acrobat Pro and Reader display page numbers in the Page Navigation toolbar. The page number format is specified by the /PageLabels entry in the Document Catalog.

Many documents use specific page number formats within a document. Commonly, front matter is numbered with lowercase Roman numerals. The main content, starting on the page numbered 1, may actually be the fifth or sixth page in the document. Similarly, appendices may begin with page number 1 plus a prefix of the appendix letter (e.g., "A-1").

Authors should make sure that the page numbering of their converted documents is reflected in any page number displays in their user agent. Consistency in presenting the document's page numbers will help make navigating the document more predictable and understandable.

As an example, if /PageLabels has not been provided to describe the page number formatting, the page numbering scheme will not be reflected in the Page Navigation toolbar in Adobe Acrobat Pro or Reader. This toolbar displays the page number in a text box, which users can change to move to another page. In addition, users can select the arrows to move one page up or down in the document. The toolbar also displays the relative page number location. In the image below, the default display indicates the user is on page 1 of 4 pages.



A more direct way of going to a page is to use the shortcut for the View > Page Navigation > Page menu item. On Windows, this shortcut is "Ctrl + Shift + N"; on Mac OS, it is "Cmd + Shift + N". This brings up a dialog box to go to a specific page number.

Examples

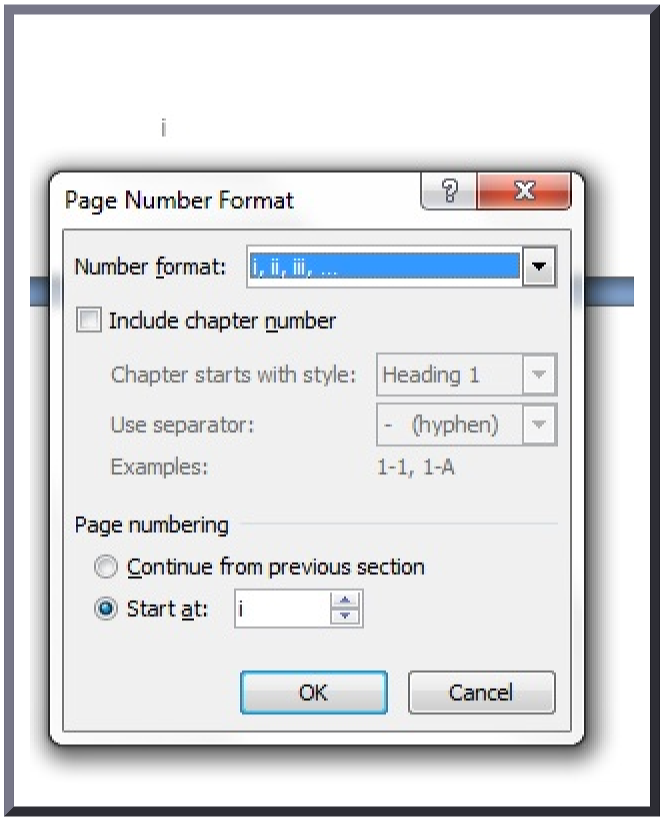
Example 1: Editing PDF page number formatting specifications using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

The example document converted from Microsoft Word 2007 has 4 pages, numbered i, ii, iii, 1. The image below shows the Word document with lowercase Roman numeral page numbering specified In Word using:

- Insert ribbon > Page number > Page Number Format

In this document, a new section has been created with page numbering beginning with Arabic numeral 1 on the fourth page of the document. The document was then converted to PDF from Word.



i

Page Number Format

Number format: i, ii, iii, ...

☐ Include chapter number

Chapter starts with style: Heading 1

Use separator: - (hyphen)

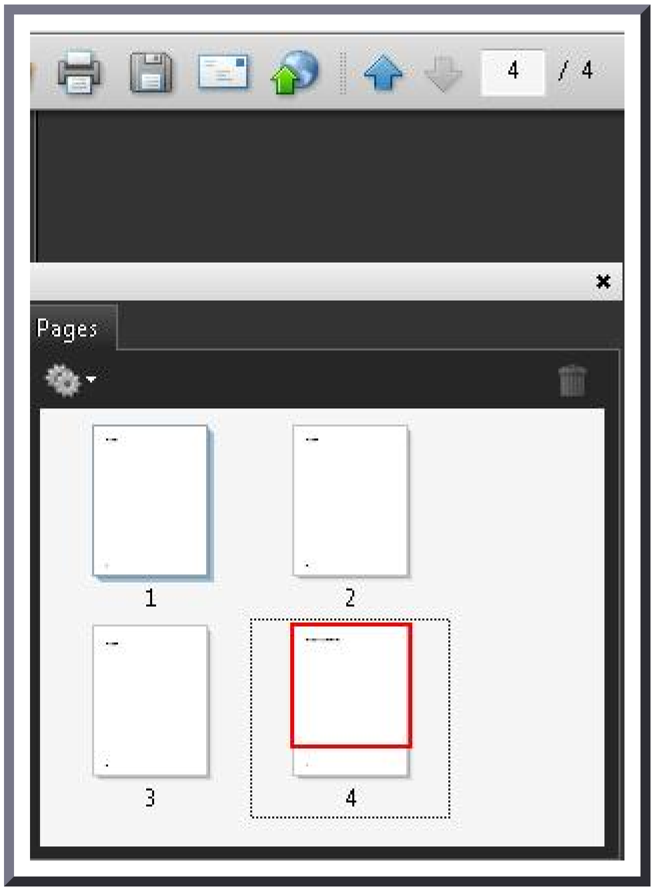
Examples: 1-1, 1-A

Page numbering

○ Continue from previous section

● Start at: i

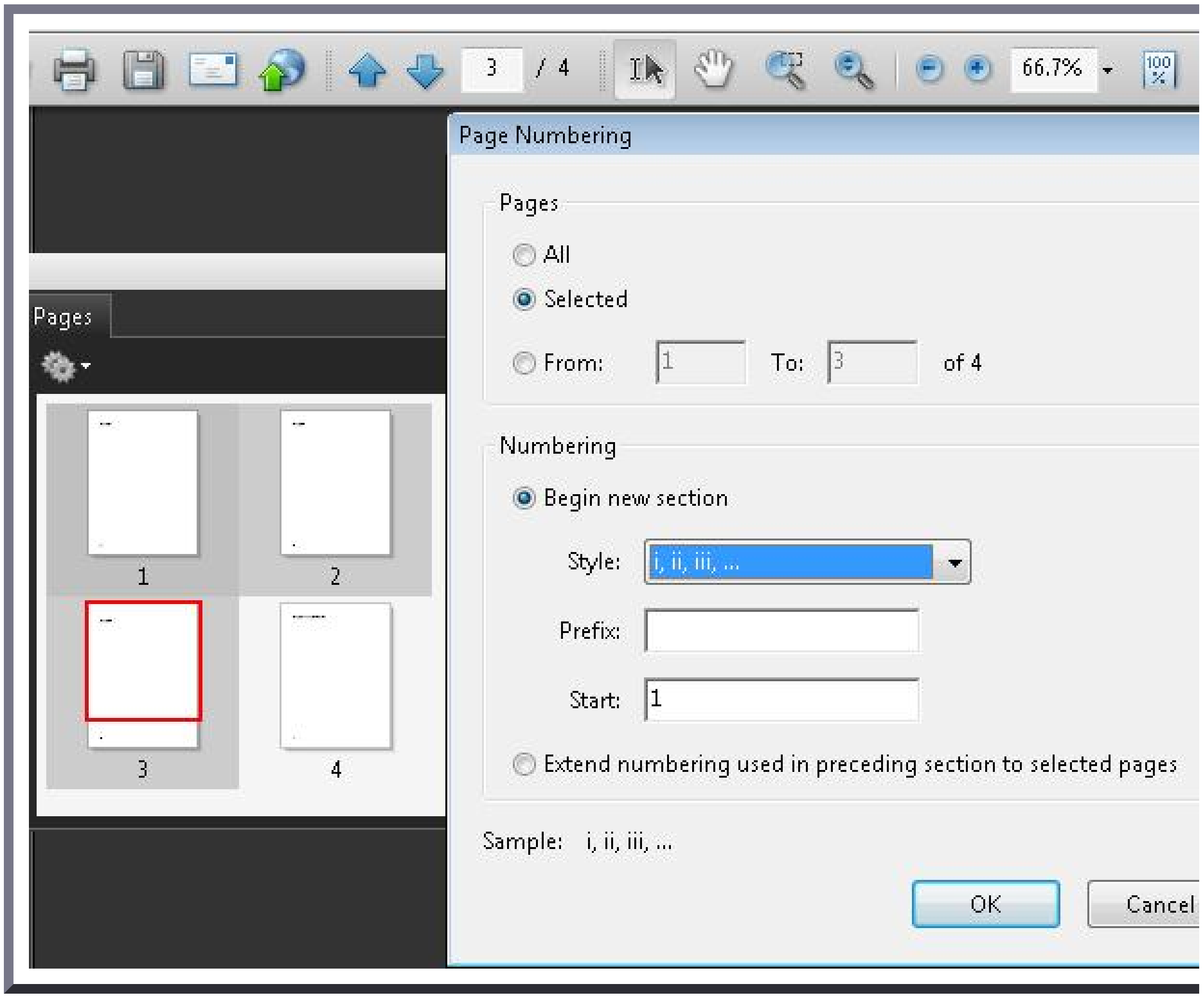
In Adobe Acrobat Pro, Select View > Navigation Panels > Pages. The following image shows the page thumbnails in the Pages panel and the Page Navigation toolbar. Both the thumbnails and the toolbar use Arabic page numbers.



To correct the page numbers:

1. Select the pages to be renumbered
2. Access the context menu for the selected pages and select Number Pages
3. In the Page Numbering dialog, select the lowercase Roman numeral style and the starting page (1 by default, which is correct in this case)
4. Select OK

The following image shows the Page Numbering dialog and selections.


Page Numbering

Pages

All
[✓ Selected]
From: 1 To: 3 of 4

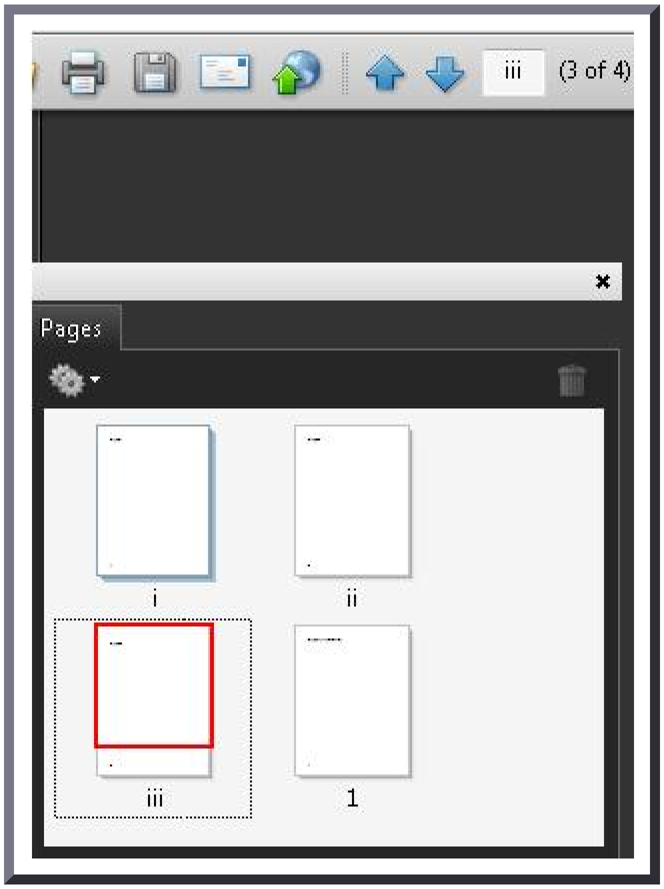
Numbering

[✓ Begin new section]
Style: i, ii, iii, ...
Prefix:
Start: 1
Extend numbering used in preceding section to selected pages

Sample: i, ii, iii, ...

Follow the same process to change the fourth page number to Arabic numeral 1.

The following image shows the correct page numbers for the 4 pages. Note that page iii is selected in the Pages panel and the Page Navigation toolbar shows iii in the text area. In addition, the relative location in the document is shown at the right of the toolbar: "(3 of 4)."



This example is shown in operation in the working example of specifying page numbers in a document converted from Word (Word file) and working example of specifying page numbers

Example 2: Specifying page numbers using the /PageLabels entry

The following code fragment illustrates code that is typical for specifying multiple page numbering schemes in a document.

The example below is for a document with pages labeled:

> *Example:* i, ii, iii, iv, 1, 2, 3, A-8, A-9, …

This numbering scheme requires 3 page-label dictionaries (for lowercase Roman, Arabic, and prefixed numbers)

```
1 0 obj
    << /Type /Catalog
       /PageLabels << /Nums [ 0 << /S /r >>  % lowercase Roman numerals
                              4 << /S /D >>  % Arabic numerals
                              7 << /S /D     % Arabic numerals with ...
                      /P (A-)                % the prefix "A-"...
                      /St 8                  % starting at page 8
                                >>
                            ]
                    >>
       …
   >>
  endobj
```

Page labels are specified as follows:

- `/S` specifies the numbering style for page numbers:
    - `/D` - Arabic numerals (1,2,3...)
    - `/r` - lowercase Roman numerals (i, ii, iii,...)
    - `/R` - uppercase Roman numerals (I, II, III,...)
    - `/A` - uppercase letters (A-Z)
    - `/a` - lowercase letters (a-z)
- `/P` (optional) - page number prefix
- `/St` (optional) - the value of the first page number in the range (default: 1)

Resources

Resources are for information purposes only, no endorsement implied.

- Section 12.4.2 (Page Labels) PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Related Techniques

- PDF14: Providing running headers and footers in PDF documents

Tests

Procedure

1. For every section in the document that uses a different pagination format, check that the page navigation feature uses the same format used on the document pages:
   - Select the pages that begin a new pagination format and visually verify that the same format and page number is shown in the page navigation feature.
   - Using a screen reader, check that the page number announced in the page navigation feature is the same as the page number announced on the document page.
   - Using a tool that is capable of showing the /PageLabels entries, open the PDF document and view the entries.
   - Use a tool that exposes the document through the accessibility API, and verify that the /PageLabels entries are specified correctly.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

PDF18: Specifying the document title using the Title entry in the document information dictionary of a PDF document

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 2.4.2 (Page Titled)
  - How to Meet 2.4.2 (Page Titled)
  - Understanding Success Criterion 2.4.2 (Page Titled)

  Note: This technique must be combined with other techniques to meet SC 2.4.2. See Understanding SC 2.4.2 for details.

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF18. Also see PDF Technology Notes.

Description

The intent of this technique is to show how a descriptive title for a PDF document can be specified for assistive technology by using the /Title entry in the document information dictionary and by setting the DisplayDocTitle flag to True in a viewer preferences dictionary. This is typically accomplished by using a tool for authoring PDF.

Document titles identify the current location without requiring users to read or interpret page content. User agents make the title of the page easily available to the user for identifying the page. For instance, a user agent may display the page title in the window title bar or as the name of the tab containing the page.

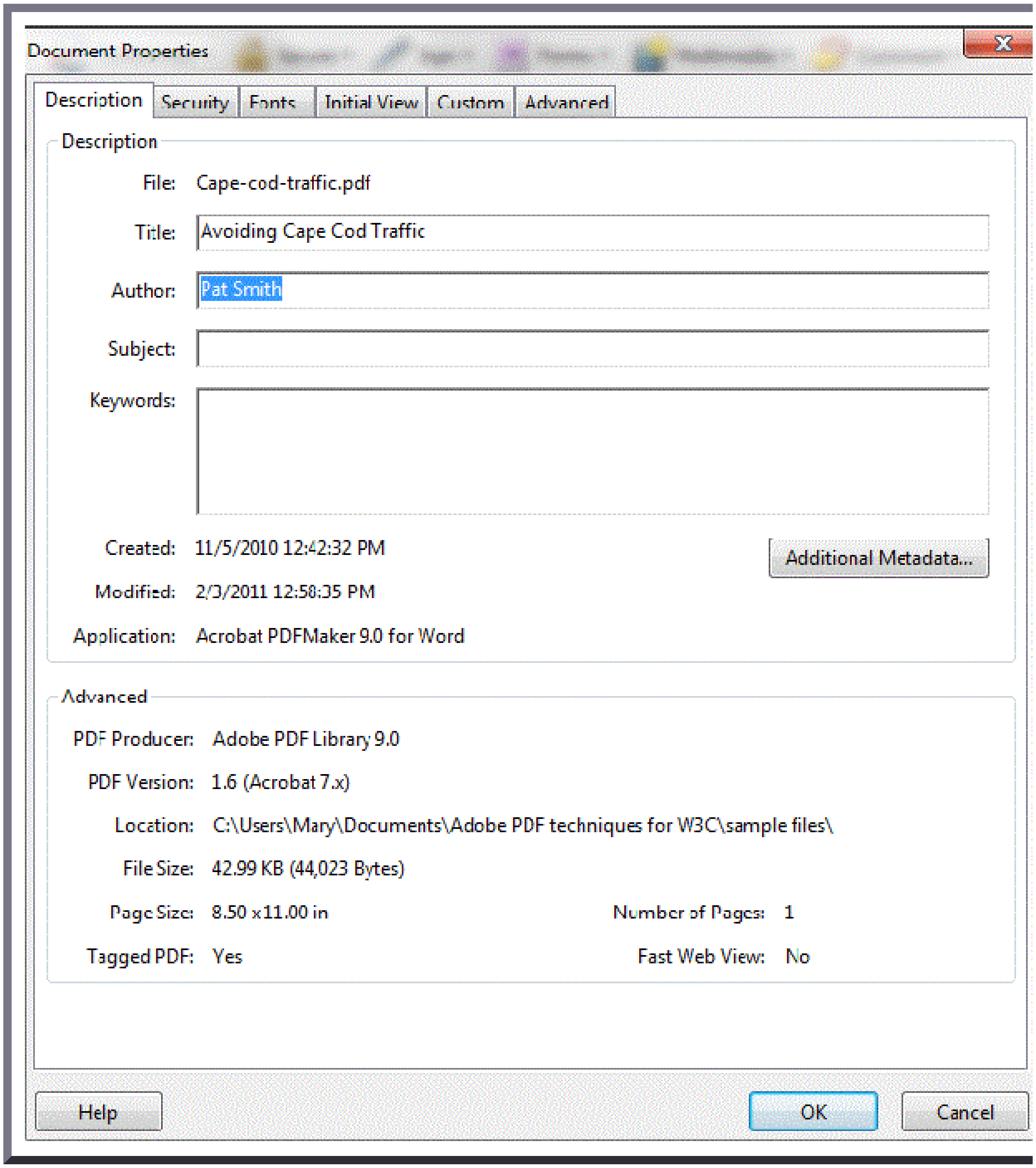
Examples

Example 1: Setting the document title in the metadata and specifying that the title be displayed in the title bar using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Open the PDF document in Adobe Acrobat Pro:

1. Select File > Properties
2. Select the Description tab to view the metadata in the document, including the document information dictionary
3. Modify the Title field to add or change the document's Title entry



Description

File: Cape-cod-traffic.pdf

Title: Avoiding Cape Cod Traffic

Author: Pat Smith

Subject:

Keywords:

Created: 11/5/2010 12:42:32 PM

Modified: 2/3/2011 12:58:35 PM

Application: Acrobat PDFMaker 9.0 for Word

Advanced

PDF Producer: Adobe PDF Library 9.0

PDF Version: 1.6 (Acrobat 7.x)

Location: C:\Users\Mary\Documents\Adobe PDF techniques for W3C\sample files\

File Size: 42.99 KB (44,023 Bytes)

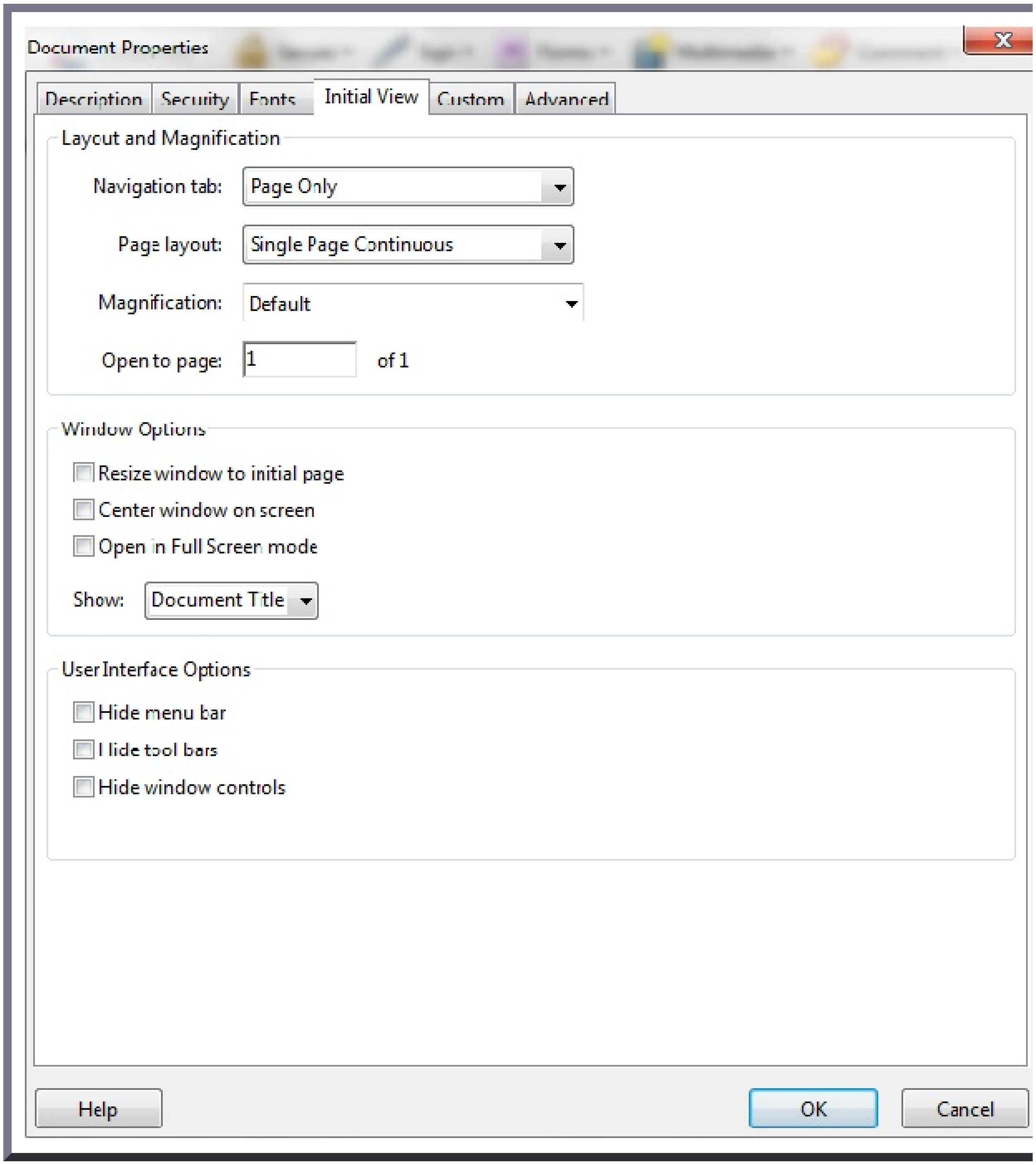
Page Size: 8.50 x 11.00 in Number of Pages: 1

Tagged PDF: Yes Fast Web View: No

Note that, with Adobe Acrobat installed, you can also enter and read the data properties information from the desktop. Access the file's context menu, choose Properties, and select the PDF tab. Any information you type or edit in this dialog box also appears in the Document Properties Description when you open the file.

To display the document title in the title bar of a user agent:

1. Select File > Properties
2. Select the Initial View tab
3. In the Window Options section, select Document Title in the Show pull-down list.



Document Properties

Description | Security | Fonts | Initial View | Custom | Advanced

Layout and Magnification

Navigation tab: Page Only

Page layout: Single Page Continuous

Magnification: Default

Open to page: 1 of 1

Window Options

- [ ] Resize window to initial page
- [ ] Center window on screen
- [ ] Open in Full Screen mode

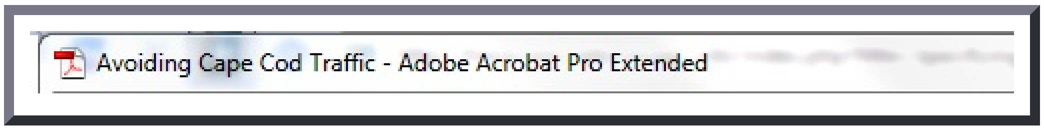
Show: Document Title

User Interface Options

- [ ] Hide menu bar
- [ ] Hide tool bars
- [ ] Hide window controls

Help OK Cancel

The title is displayed in the title bar, as shown in the image below.



Avoiding Cape Cod Traffic - Adobe Acrobat Pro Extended

This example is shown in operation in the [working example of displaying document title in the](https://www.w3.org/TR/WCAG20-TECHS/pdf)

Example 2: A /Title entry in the document information dictionary of a PDF document

The following code fragment illustrates code that is typical for providing a /Title entry in a document information dictionary that contains a document title.

```
1 0 obj
   << /Title (Applying Guerrilla Tactics to Usability Testing by People with Disabilities)
      /Author (Mary Smith)
      /CreationDate (D:19970915110347-08'00')
   >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- PDF and Accessibility
- Section 14.3.3 (Document Information Dictionary) in PDF 1.7 (ISO 32000-1)
- PDF Reference 1.6, TITLE entry of the document information dictionary

Related Techniques

- G88: Providing descriptive titles for Web pages

Tests

Procedure

1. Verify that the title for the document is correctly specified and displayed in the user agent title bar by applying one of the following:
   - Open the PDF document with a screen reader, listening to hear that the document title is read correctly.
   - Using a PDF editor, check that the document title is specified. Select the Initial View tab to check that the title will be displayed.

  - Use a tool which is capable of showing the /Title entry value in the document catalog to open the PDF document and view the /Title entry and /DisplayDocTitle flag settings.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF19: Specifying the language for a passage or phrase with the Lang entry in PDF documents

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 3.1.1 (Language of Page)
  - How to Meet 3.1.1 (Language of Page)
  - Understanding Success Criterion 3.1.1 (Language of Page)
- Success Criterion 3.1.2 (Language of Parts)
  - How to Meet 3.1.2 (Language of Parts)
  - Understanding Success Criterion 3.1.2 (Language of Parts)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF19. Also see PDF Technology Notes.

Description

The objective of this technique is to specify the language of a passage, phrase, or word using the /Lang entry to provide information in the PDF document that user agents need to present text and other linguistic content correctly. This is normally accomplished using a tool for authoring PDF.

Both assistive technologies and conventional user agents can render text more accurately when the language is identified. Screen readers can load the correct pronunciation rules. As a result, users with disabilities are better able to understand the content.

> *Note:* This technique can be used to set the default language for the entire document if the entire document is contained in the container or tag. In this case, this technique would apply to Success Criterion 3.1.1.

Examples

Example 1: Adding a /Lang entry to specify the language for a paragraph using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. In the Tools menu, select Advanced Editing.
2. Select the TouchUp Reading Order Tool.
3. Click the Show Order Panel button in the TouchUp Reading Order Tool
4. Select the Tags tab in the Show Order Panel and select the paragraph that is in the different language. You can also use the Options menu in the Tags tab: select Find Tag from Selection.
5. Right-click the selection and select Properties in the context menu.
6. In the Tags tab in the Properties dialog, select the language from the drop-down list.

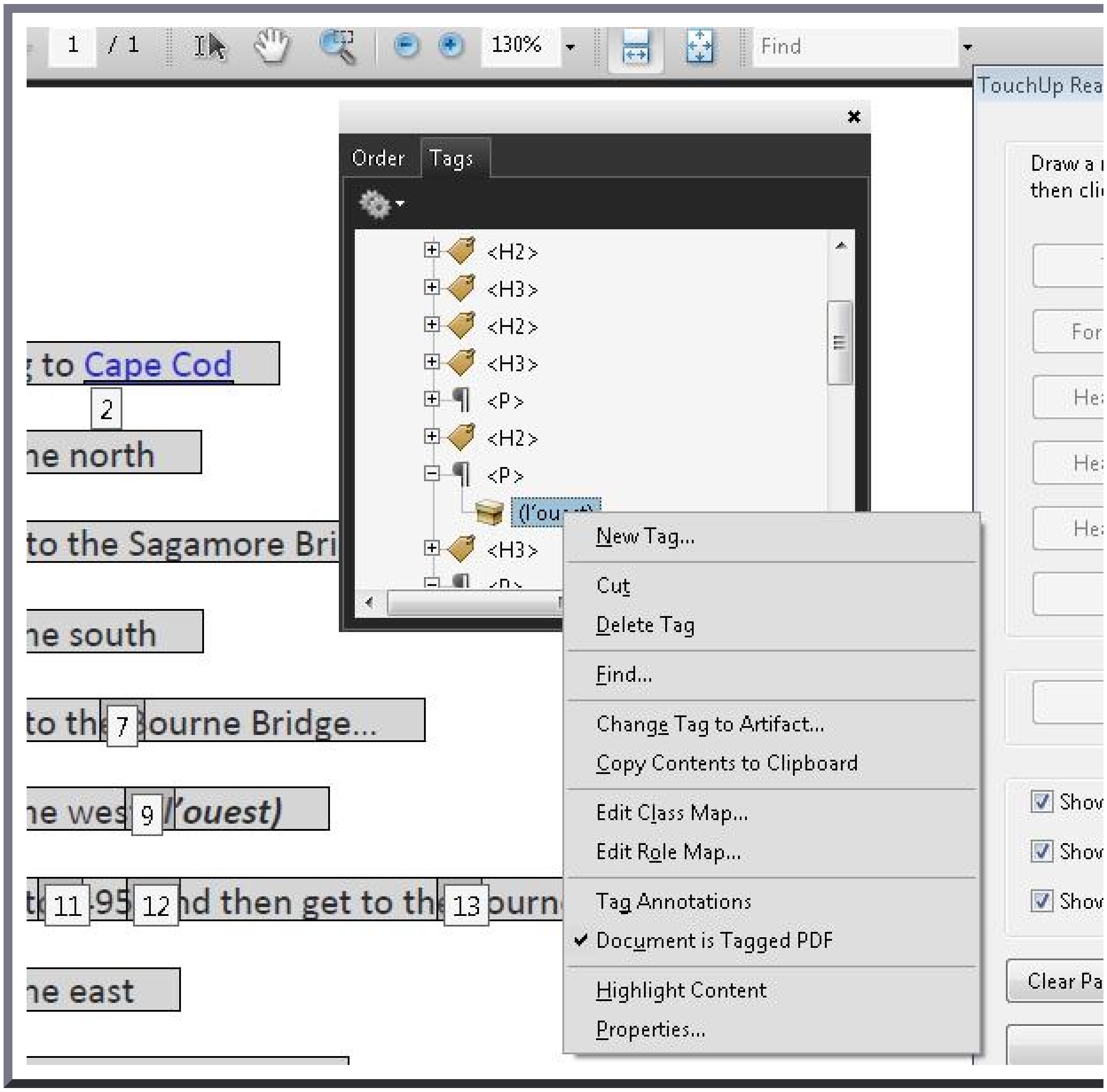
> *Note:* Acrobat includes 16 preset language selections. If you need to specify a language that is not on the list, such as Russian, you must type the ISO 639 code for the language, not its name.

Example 2: Adding a /Lang entry to specify the language for a specific word or phrase using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Select the word or phrase that is in a different language and create a tag for it in the Reading Order Panel (e.g., Text).
2. Open the Tags tab in the Show Order Panel and select the tagged word or phrase that is in the different language. You can also use the Options menu in the Tags tab: select Create Tag from Selection.
3. Right-click the selection and select Properties in the context menu.
4. In the Tags tab in the Properties dialog, select the language from the drop-down list.

When you tag a word or phrase, Acrobat splits the original content into three document content tags: one for the text that precedes your selection, one for the selection, and one for the text that follows the selection. As needed, drag the document content tag for the selected text into position between the other two tags, so that the text reads in the proper order. All three tags must also be at the same level beneath their parent tag. Drag them into place if they are not.



This example is shown in operation in the [working example of marking a specific word or phrase in Acrobat Pro]

Example 3: Specifying the language for a word or phrase in a PDF document using a /Lang entry

Below the level of the default document language, the language for a passage may be specified for the following items:

- Marked-content sequences that are not in the structure hierarchy, through a /Lang entry in a property list attached to the marked-content sequence with a Span tag.
- Structure elements of any type, through a /Lang entry in the structure element dictionary.

The following code fragment illustrates code that is typical for using the /Lang entry to override the default document language by specifying a marked-content sequence within a page's content stream:

```
/P % Start of marked-content sequence
BDC
   (See you later, or in Spanish you would say, ) Tj
   /Span << /Lang (es-MX) >>% Start of nested marked-content sequence
  BDC
   (Hasta la vista.) Tj
  EMC% End of nested marked-content sequence
EMC% End of marked-content sequence
```

The following code fragment illustrates code that is typical for using the /Lang entry in the structure element dictionary. In this case, the /Lang entry applies to the marked-content sequence having an MCID (marked-content identifier) value of 0 within the indicated page's content stream.

```
1 0 obj% Structure element
  << /Type /StructElem
    /S /Span% Structure type
    /P /P% Parent in structure hierarchy
    /K<< /Type /MCR
      /Pg 2 0 R% Page containing marked-content sequence
      /MCID 0% Marked-content identifier
     >>
   /Lang (es-MX)% Language specification for this element
   >>
endobj
2 0 obj% Page object
  << /Type /Page
     /Contents 3 0 R% Content stream
   …
   >>
   endobj
3 0 obj% Page's content stream
```

```
 << /Length … >>
   stream
    BT
     /P % Start of marked-content sequence
     BDC
    (See you later, or in Spanish you would say, ) Tj
    /Span << /MCID 0 >>% Start of nested marked-content sequence
   BDC
    (Hasta la vista.) Tj
   EMC% End of nested marked-content sequence
 EMC% End of marked-content sequence
ET
endstream
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.9.2 (Natural Language Specification) in PDF 1.7 (ISO 32000-1)
- ISO 639-2 Codes
- PDF Reference 1.6, 10.8.1 Natural Language Specification (PDF 8.7 Mb)
- PDF Standards: Natural Language Specification
- Adobe® Acrobat® 9 Pro Accessibility Guide: Creating Accessible PDF from Microsoft® Word
- PDF and Accessibility

Related Techniques

- PDF16: Setting the default language using the /Lang entry in the document catalog of a PDF document

Tests

Procedure

1. Verify that the language of a passage, phrase, or word that differs from the language of the surrounding text is correctly specified by a /Lang entry on an enclosing tag or container:

  - Read the PDF document with a screen reader that supports the language of the phrase and the language of the surrounding text, listening to hear that the text is read in the correct natural language.
  - Using a PDF editor, select the word or phrase that is in the different language and check that the language is set correctly.
  - Use a tool which is capable of showing the /Lang entry value to open the PDF document and view the language settings.
  - Use a tool that exposes the document through the accessibility API and verify that the language for the passage or phrase is set correctly.
2. Verify that if the container or tag contains the entire document, the language setting is the language intended as the default for the document.

Expected Results

- #1 and #2 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF20: Using Adobe Acrobat Pro's Table Editor to repair mistagged tables

Applicability

Tagged PDF documents with tables.

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
  - How to Meet 1.3.1 (Info and Relationships)
  - Understanding Success Criterion 1.3.1 (Info and Relationships)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF20. Also see PDF Technology Notes.

Description

The purpose of this technique is to show how table cells in PDF documents can be marked up so that the logical relationships among rows and columns are preserved and recognized by assistive technology. This is typically accomplished by using a tool for authoring PDF.

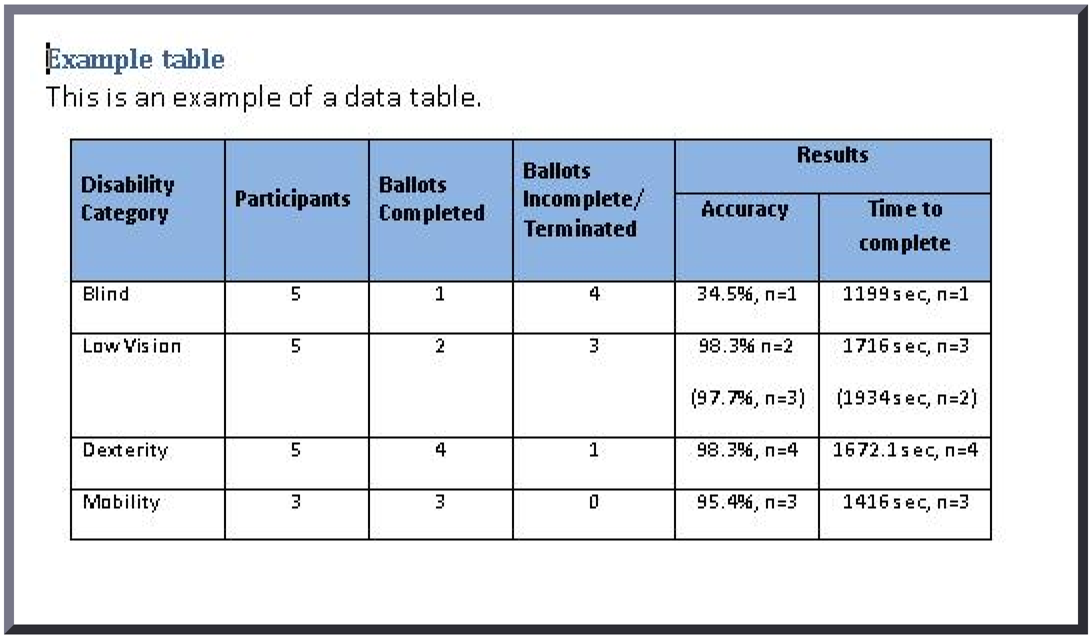
However, tables converted to PDF may have incorrectly merged or split table cells, even if they were marked up correctly in the authoring tool. Authors can ensure that table cells are structured properly by using the Table Editor in Adobe Acrobat Pro's TouchUp Reading Order tool.

Examples

Example 1: Repairing table cells using the Table Editor in the TouchUp Reading Order tool in Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

This example uses a table that was marked up correctly when it was created in Microsoft Word. Some table headers span two rows in the header row; one table header spans two columns.



Example table

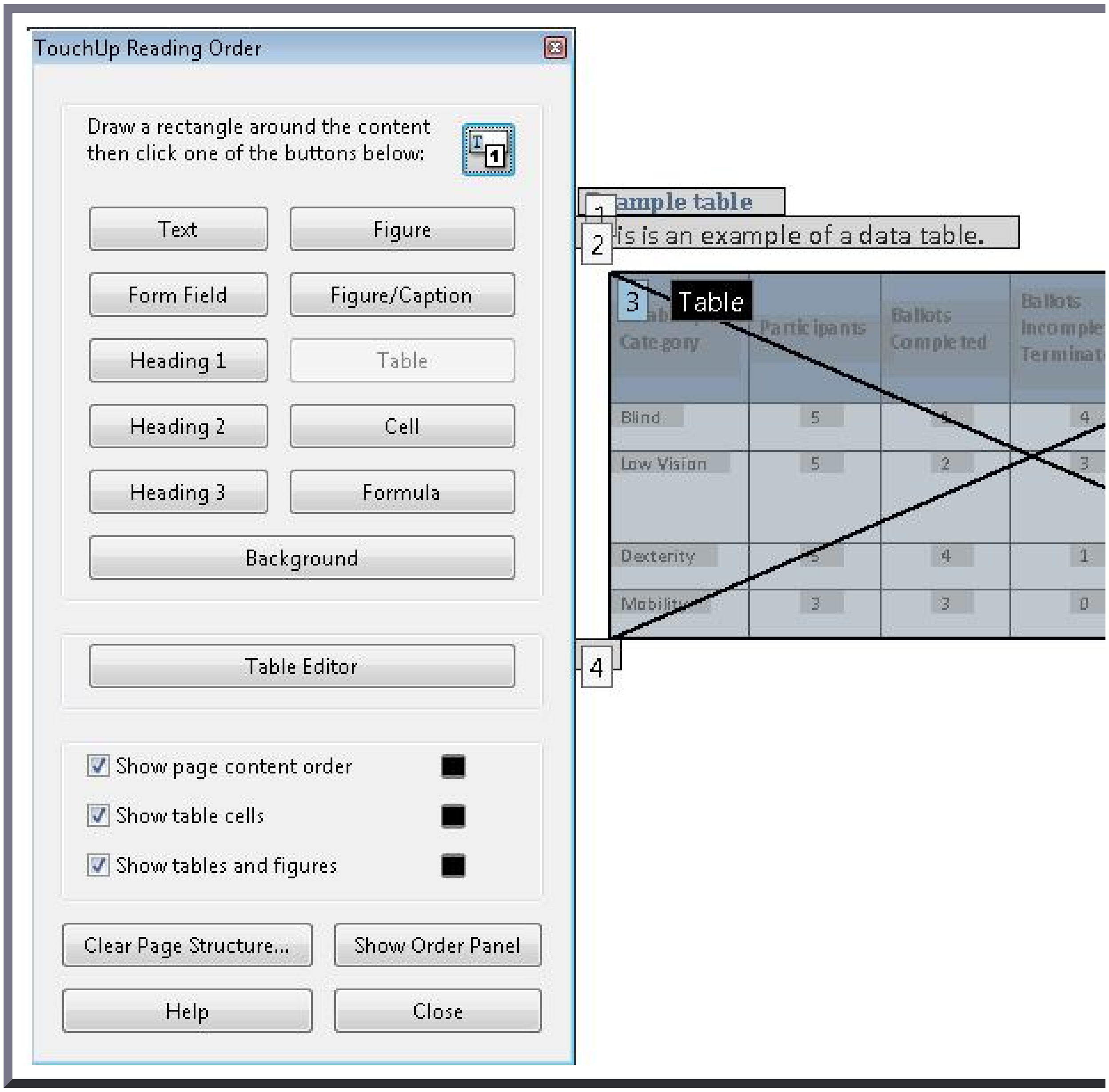
This is an example of a data table.

Disability Category	Participants	Ballots Completed	Ballots Incomplete/ Terminated	Results	
				Accuracy	Time to complete
Blind	5	1	4	34.5%, n=1	1199 sec, n=1
Low Vision	5	2	3	98.3% n=2 (97.7%, n=3)	1716 sec, n=3 (1934 sec, n=2)
Dexterity	5	4	1	98.3%, n=4	1672.1 sec, n=4
Mobility	3	3	0	95.4%, n=3	1416 sec, n=3

To check the table in the PDF document:

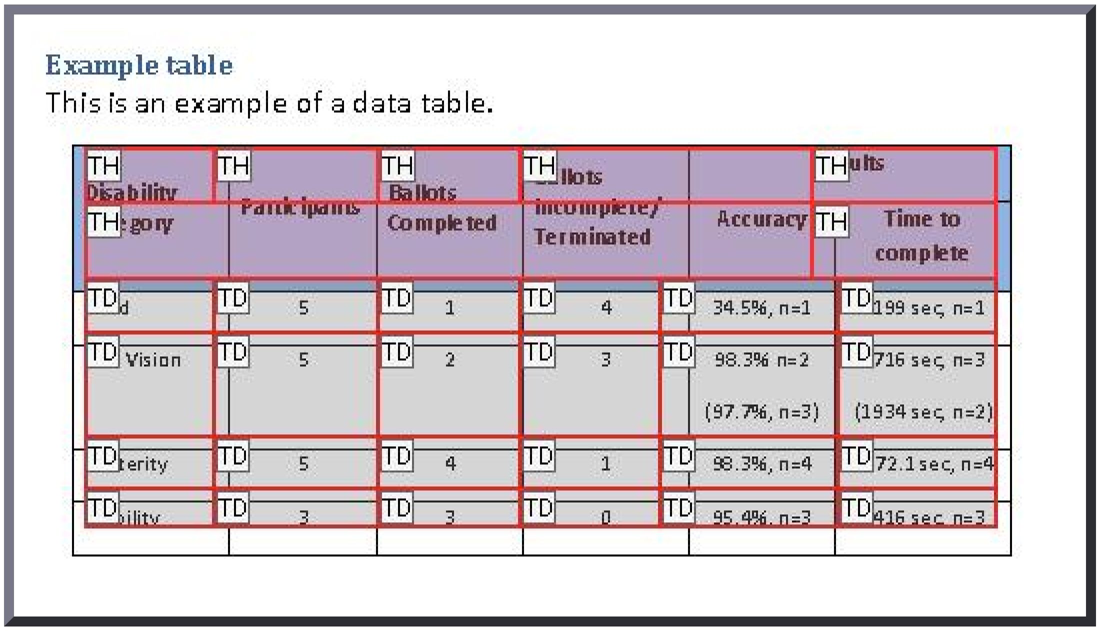
1. Advanced > Accessibility > TouchUp Reading Order...
2. Select the table by clicking the number in the top left hand corner of the table (3 in the reading order in the image below).
3. Select the Table Editor button on the TouchUp Reading Order panel. The table cells will be outlined in red and labeled with their tags. The red outlines may not exactly match up to the table cells but you should be able to determine if the cells are tagged correctly.

The following image shows the example table in the TouchUp Reading Order tool. Note that the Results header appears to span two sub-headers and the other headers to the left span the two rows in the Results header.



Category	Participants	Ballots Completed	Ballots Incomple Terminat
Blind	5	1	4
Low Vision	5	2	3
Dexterity	5	4	1
Mobility	3	3	0

The following images shows the example table in the Table Editor. The cells are outlined in red, and the tab for each cell is displayed. Upon conversion, the Results header was incorrectly split and does not span its two sub-headers. The headers to the right were incorrectly split into 2 cells each and do not span the Results headers. In addition, the incorrectly split cells were merged into one cell.



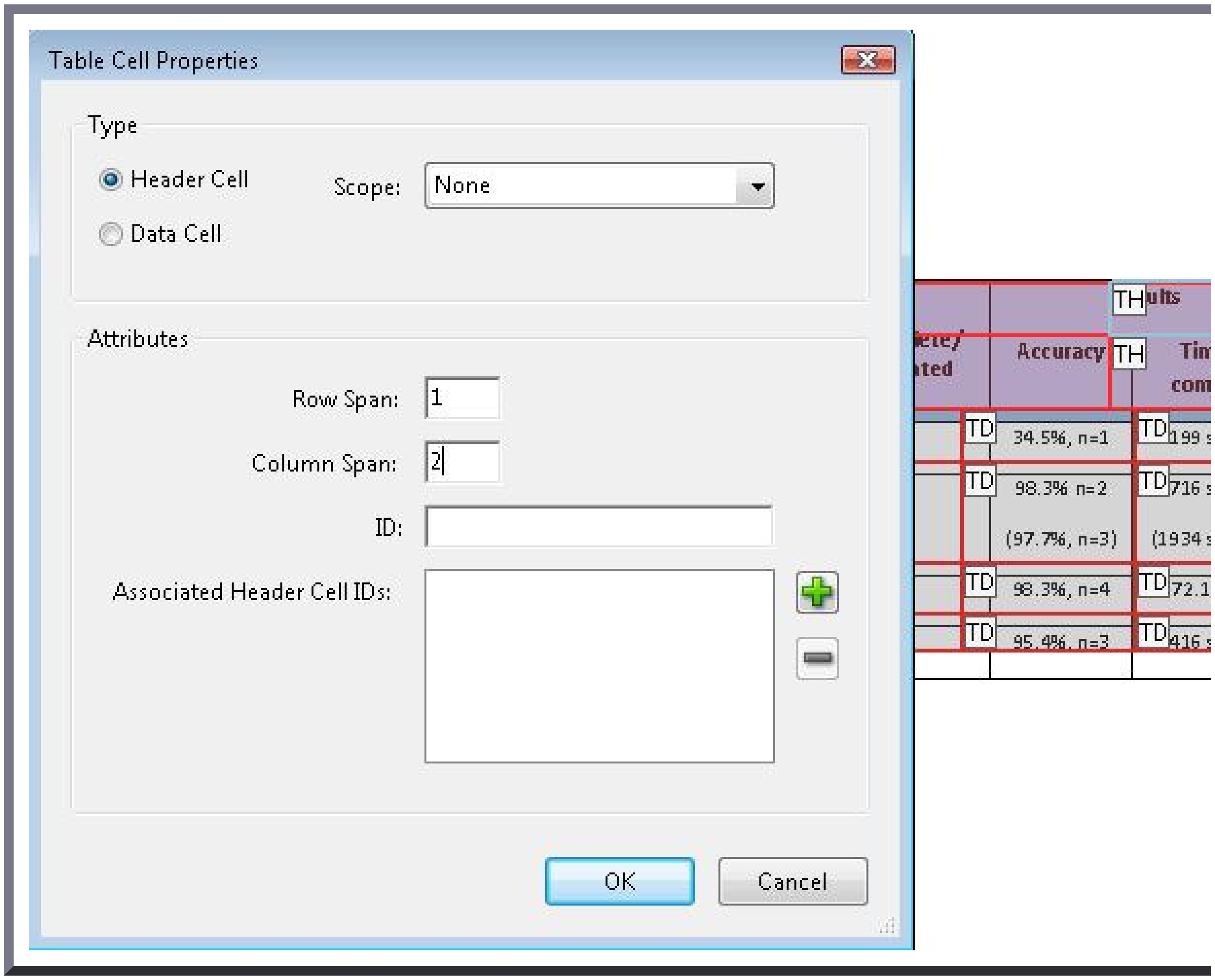
Example table

This is an example of a data table.

TH Disability TH egory	TH Participants	TH Ballots Completed	TH llots incomplete/ Terminated		TH ults
				Accuracy	TH Time to complete
TD d	TD 5	TD 1	TD 4	TD 34.5%, n=1	TD 199 sec, n=1
TD Vision	TD 5	TD 2	TD 3	TD 98.3% n=2 (97.7%, n=3)	TD 716 sec, n=3 (1934 sec, n=2)
TD terity	TD 5	TD 4	TD 1	TD 98.3%, n=4	TD 72.1 sec, n=4
TD bility	TD 3	TD 3	TD 0	TD 95.4%, n=3	TD 416 sec, n=3

To repair the Results header:

1. Select the header in the table (it will be outlined in blue when selected)
2. Access the context menu
3. Select Table Cell Properties...
4. In the Table Cell Properties dialog, change the Column Span from 1 to 2
5. Press OK. You'll get a warning that the change might result in a malformed table structure. In this case, the change is correct. The cell you changed should change color to show the new span, as shown in the following image.



Similarly, to repair the incorrectly split header cells to the left of Results header:

1. Select the top cell in the column (it will be outlined in blue when selected)
2. Access the context menu
3. Select Table Cell Properties...
4. In the Table Cell Properties dialog, change the Row Span from 1 to 2
5. Press OK. The following image shows the correction being made to the last header cell, with the corrected header cells to its left.

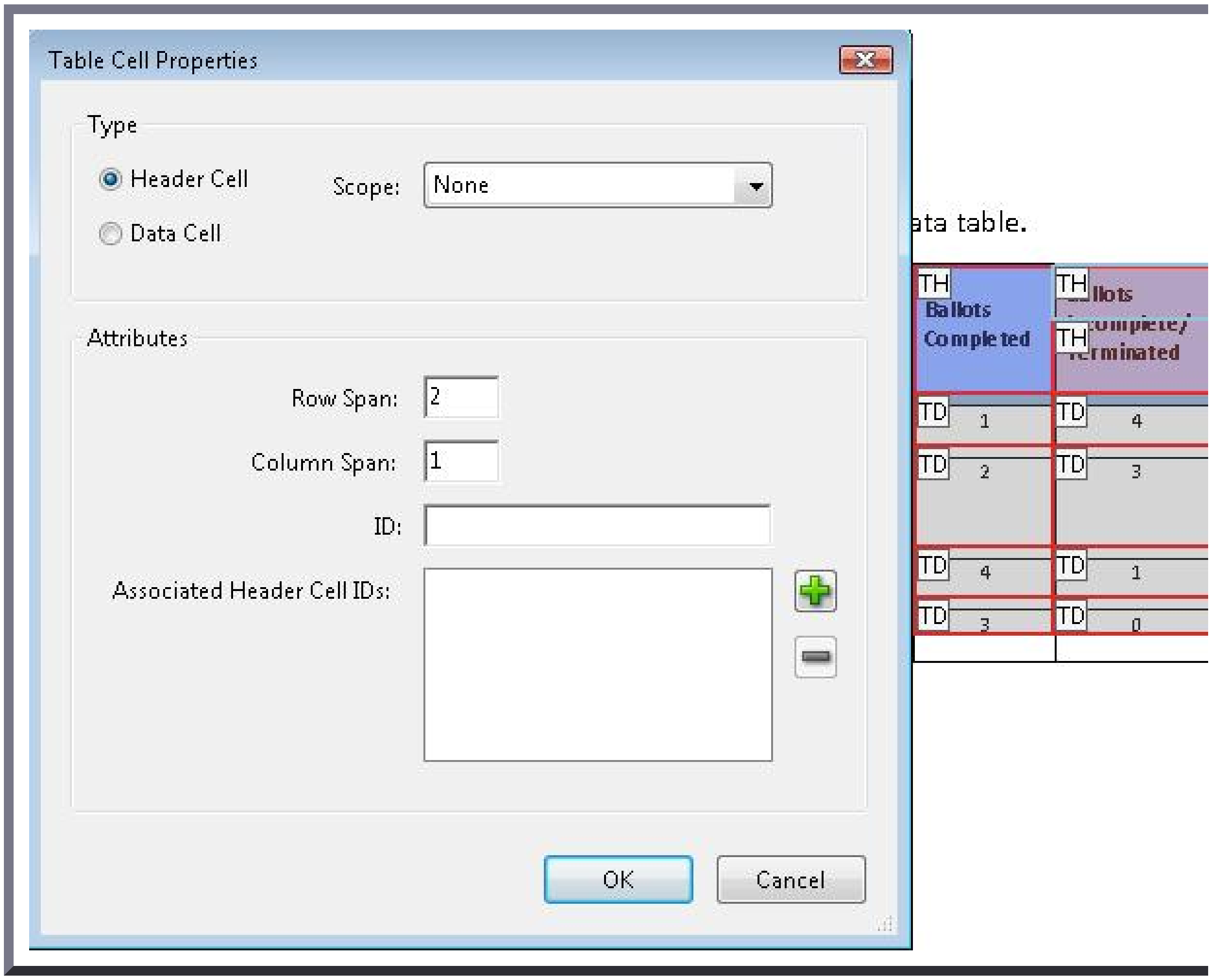


Table Cell Properties

Type

[✓ Header Cell] Data Cell

Scope: None

Attributes

Row Span: 2

Column Span: 1

ID:

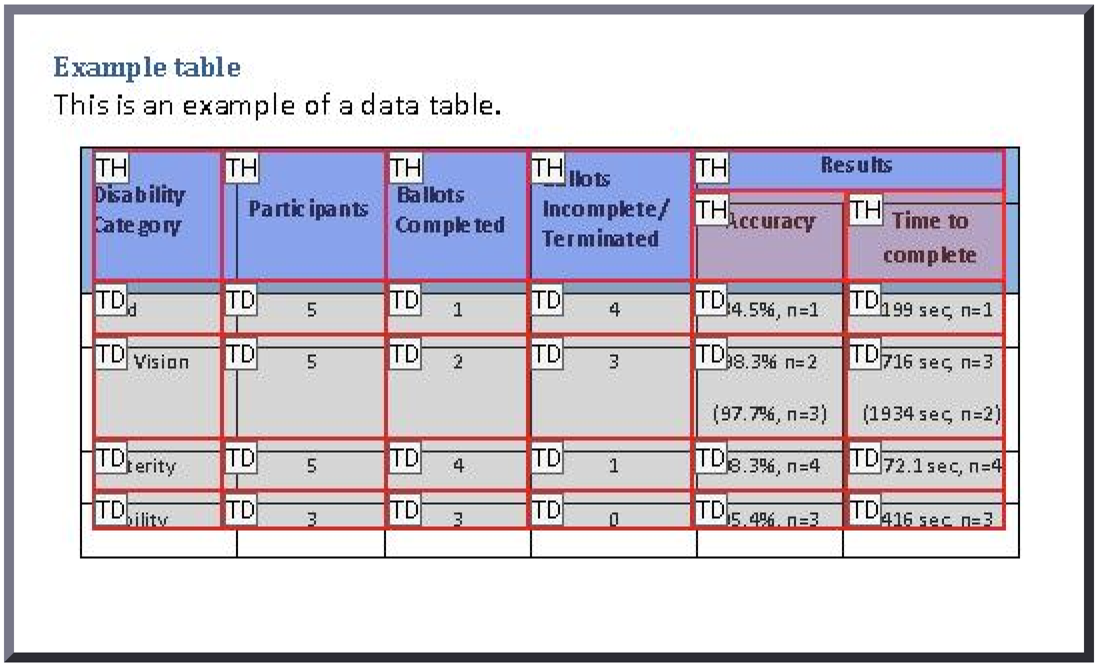
Associated Header Cell IDs:

OK Cancel

ata table.

TH Ballots Completed	TH llots omplete/ TH rminated
TD 1	TD 4
TD 2	TD 3
TD 4	TD 1
TD 3	TD 0

The following image shows the repaired example table.



Example table

This is an example of a data table.

TH Disability Category	TH Participants	TH Ballots Completed	TH llots Incomplete/ Terminated	TH Results	
				TH Accuracy	TH Time to complete
TD d	TD 5	TD 1	TD 4	TD 4.5%, n=1	TD 199 sec, n=1
TD Vision	TD 5	TD 2	TD 3	TD 38.3% n=2 (97.7%, n=3)	TD 716 sec, n=3 (1934 sec, n=2)
TD terity	TD 5	TD 4	TD 1	TD 8.3%, n=4	TD 72.1 sec, n=4
TD bility	TD 3	TD 3	TD 0	TD 5.4%, n=3	TD 416 sec, n=3

This example is shown in operation in the working example of repairing table structure (Word file) and working example of repairing table structure (PDF file).

Example 2: Marking up a table using table structure elements

The following code fragment illustrates code that is typical for a simple table (header row and data row) such as shown in Examples 1-3:

```
95 0 obj                %Structure element for a table
 <<
  /A 39 0 R
  /K[96 0 R 101 0 R 106 0 R 111 0 R]
  /P 93 0 R
  /S/Table              %standard structure type is table
 >>
 endobj
96 0 obj                %Structure element for a table row
 <<
  /K[97 0 R 98 0 R 99 0 R 100 0 R]
  /P 95 0 R
  /S/TR                 %standard structure type is table row
 >>
 endobj
97 0 obj                %Structure element for a table header
 <</A[23 0 R 120 0 R]
   /K 1
   /P 96 0 R
   /S/TH                 %standard structure type is table head
   /Pg 8 0 R
 >>
endobj
104 0 obj                %Structure element for table data (cell contents)
 <<
  /A 29 0 R
  /K 7
  /P 101 0 R
  /S/TD                  %standard structure type is table data
  /Pg 8 0 R
 >>
endobj
```

Resources

Resources are for information purposes only, no endorsement implied.

- PDF and Accessibility
- 14.8.4.3.4 (Table Elements) in PDF 1.7 (ISO 32000-1)

Related Techniques

- H51: Using table markup to present tabular information
- PDF6: Using table elements for table markup in PDF Documents

Tests

Procedure

1. For a table that has been repaired with the Table Editor, confirm one of the following:
    - Read the PDF document with a screen reader, listening to hear that the tabular information is presented in a way that preserves logical relationships among the table header and data cells. (Configure the screen reader to not use heuristics to read table header cells.)
    - Using a PDF editor, verify that the appropriate *TR*, *TH*, and *TD* tags are in the proper reading order and hierarchy in the table tree.
    - Use a tool which is capable of showing the table elements to open the PDF document, view the table structure, and verify that it contains the appropriate TR, TH, and TD structures.
    - Use a tool that exposes the document through the accessibility API, and verify that the table structure contains the appropriate TR, TH, and TD structures, and that they are in the proper reading order and hierarchy.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF21: Using List tags for lists in PDF documents

Applicability

Tagged PDF documents with lists.

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
  - How to Meet 1.3.1 (Info and Relationships)
  - Understanding Success Criterion 1.3.1 (Info and Relationships)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF21. Also see PDF Technology Notes.

Description

The intent of this technique is to create lists of related items using list elements appropriate for their purposes. PDF files containing lists are normally created or repaired using a tool for authoring PDF.

When markup is used that visually formats items as a list but does not indicate the list relationship, users may have difficulty navigating the information. An example of such visual formatting is simply using line-breaks to separate list items.

Some assistive technologies allow users to navigate from list to list or item to item. If the lists are not correctly formatted with list tags, these users will have difficulty understanding the list content.

The easiest way to create lists in PDF content is to format them properly using list markup in the authoring tool, for example, Microsoft Word or OpenOffice.org Writer. However, if you do not have access to the source file and authoring tool, you can use Acrobat Pro's TouchUp Reading Order tool and the Tags panel.

The PDF specification defines list structure in section 14.8.4.3.3 (List Elements). The structure types for lists in PDF documents are:

- L - the List tag, which contains one or more LI tags.
- LI - the List Item tag. List item tags can contain Lbl and LBody tags.
- Lbl - the list item label. Contains distinguishing information such as a item number or bullet character.
- LBody - the list item body. Contains list item content, or in the case of a nested list, it may contain additional List tag trees.

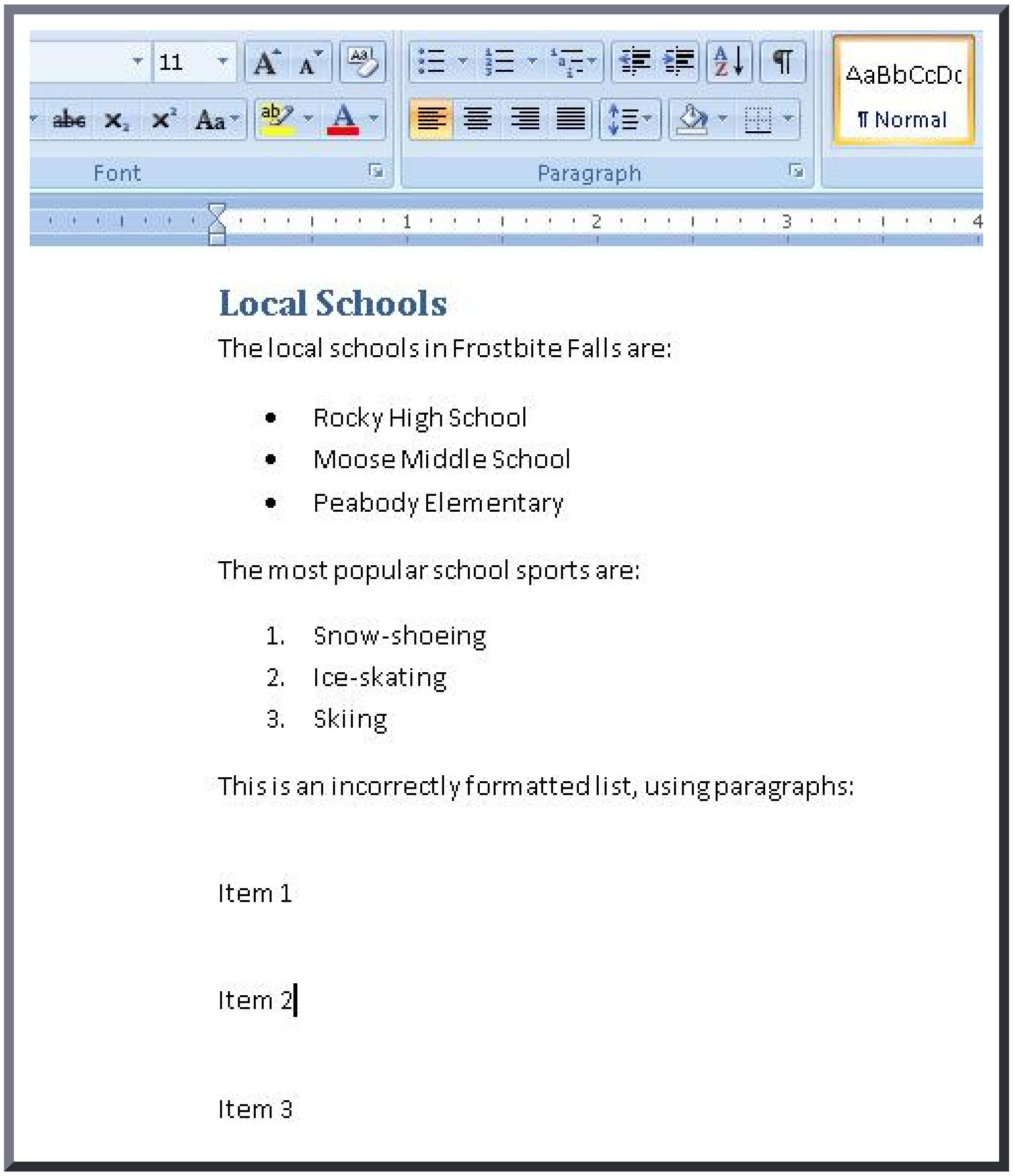
Examples

Example 1: Adding lists to Microsoft Word 2007 documents

This example is shown with Microsoft Word. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

On the Home ribbon, use the lists tools to create or repair lists in Word documents. This is the easiest way to ensure that lists are formatted correctly when they are converted to PDF.

In the image below, the numbered and bullet lists were created using the list tools. The third list did not use the list tool (see the ribbon) and the list will not be tagged as list elements when converted to PDF.



Local Schools

The local schools in Frostbite Falls are:

- Rocky High School
- Moose Middle School
- Peabody Elementary

The most popular school sports are:

1. Snow-shoeing
2. Ice-skating
3. Skiing

This is an incorrectly formatted list, using paragraphs:

Item 1

Item 2

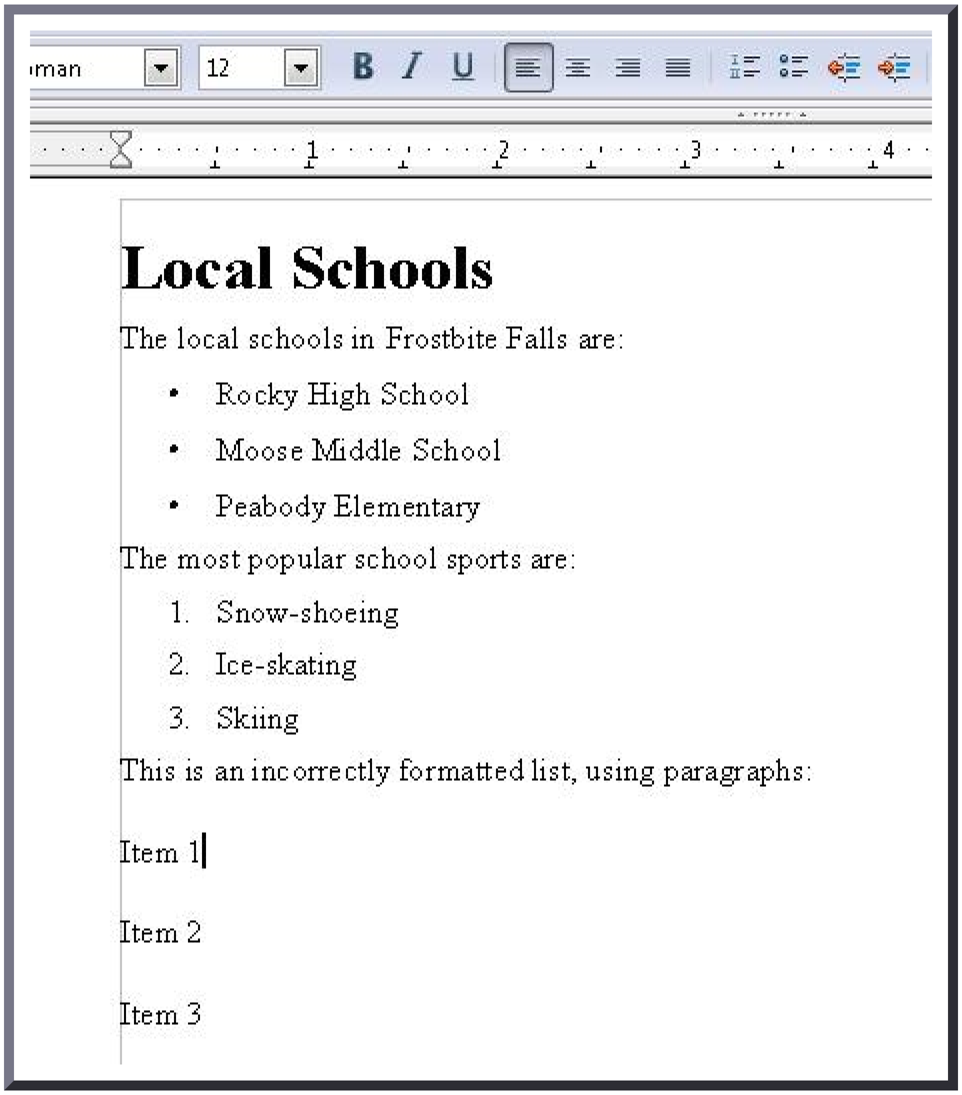
Item 3

Example 2: Adding lists to OpenOffice.org Writer 2.2 documents

This example is shown with OpenOffice.org Writer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Use the Bullets and Numbering tool to create or repair lists in OpenOffice.org Writer documents. This is the easiest way to ensure that lists are formatted correctly when they are converted to PDF.

In the image below, the numbered and bullet lists were created using the list tools. The third list did not use the list tool (see the toolbar) and the list will not be tagged as list elements when converted to PDF.



Local Schools

The local schools in Frostbite Falls are:

- Rocky High School
- Moose Middle School
- Peabody Elementary

The most popular school sports are:

1. Snow-shoeing
2. Ice-skating
3. Skiing

This is an incorrectly formatted list, using paragraphs:

Item 1

Item 2

Item 3

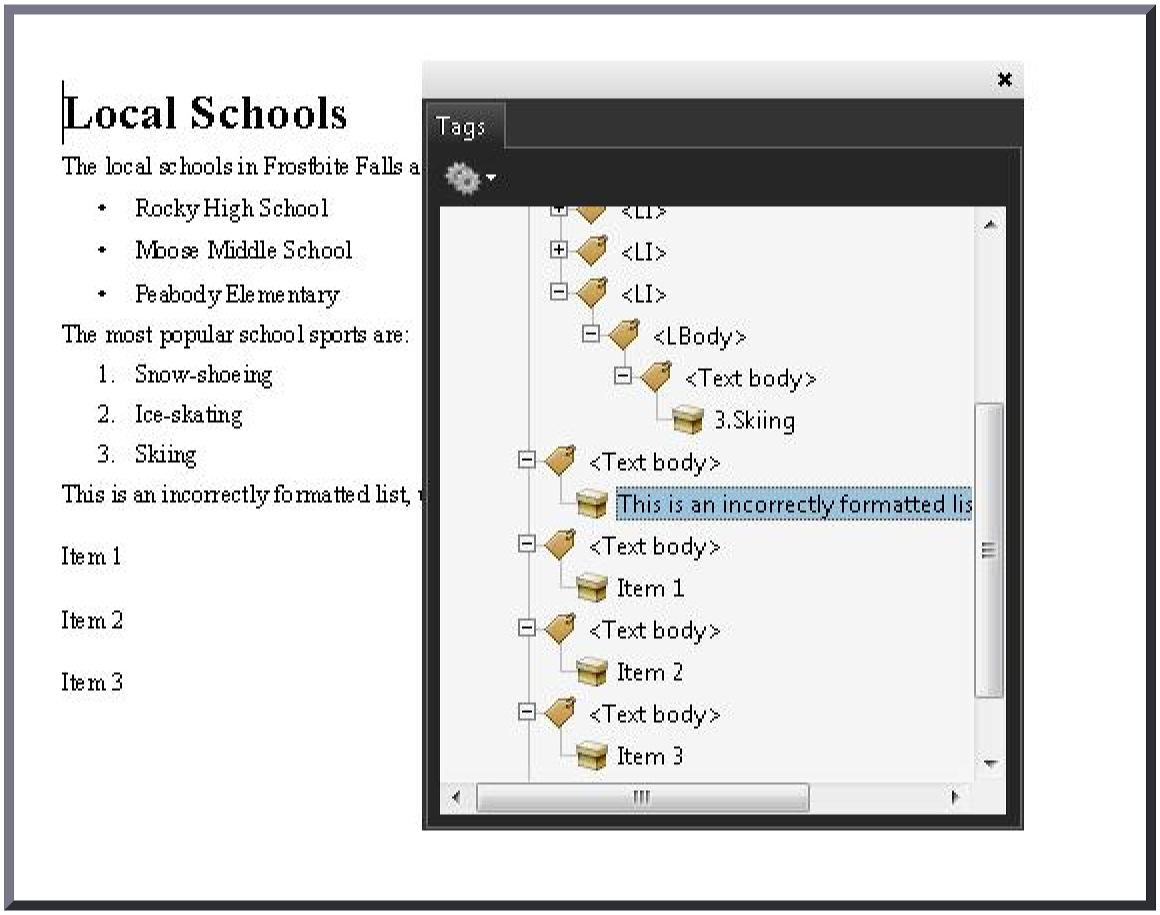
This example is shown in operation in the working example of adding lists to OpenOffice Writer documents.

Example 3: Ensuring that lists are correctly formatted using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. View > Navigation Panels... > Tags
2. Inspect the lists in the document to determine which, if any, are not formatted properly.

In the following image, the third list is formatted as text. The list items are separated only by line-breaks. Assistive technology may not be able to render the list intelligibly for users.


Local Schools

The local schools in Frostbite Falls a

- Rocky High School
- Moose Middle School
- Peabody Elementary

The most popular school sports are:

1. Snow-shoeing
2. Ice-skating
3. Skiing

This is an incorrectly formatted list,

Item 1

Item 2

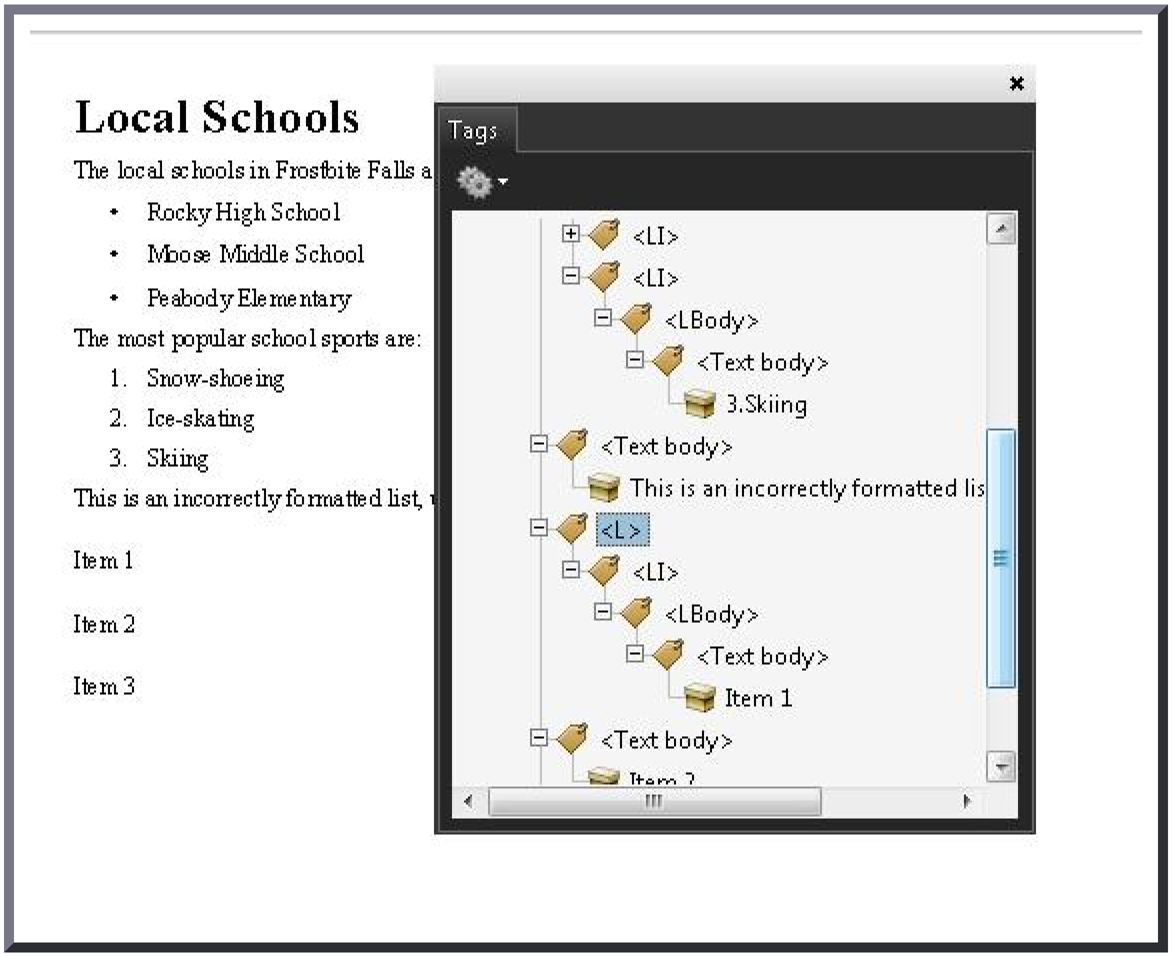
Item 3

Tags

```
<LI>
<LI>
<LI>
  <LBody>
    <Text body>
      3.Skiing
<Text body>
  This is an incorrectly formatted lis
<Text body>
  Item 1
<Text body>
  Item 2
<Text body>
  Item 3
```

To repair the list, use the Tags panel to create list tags in the content.

The following image shows the resulting first list item correctly formatted.



Local Schools

The local schools in Frostbite Falls a

- Rocky High School
- Moose Middle School
- Peabody Elementary

The most popular school sports are:

1. Snow-shoeing
2. Ice-skating
3. Skiing

This is an incorrectly formatted list, u

Item 1

Item 2

Item 3

Tags

```
<LI>
<LI>
  <LBody>
    <Text body>
      3.Skiing
<Text body>
  This is an incorrectly formatted lis
<L>
  <LI>
    <LBody>
      <Text body>
        Item 1
<Text body>
  Item 2
```

This example is shown in operation in the [working example of ensuring lists are properly formatted in Acrobat Pro](#).

Example 4: Marking up lists using List structure elements

The following code fragment illustrates code that is typical marking up a list hierarchy in PDF documents. It uses the simple numbered list in the previous examples. This is typically accomplished by an authoring tool.

```
4 0 obj
  <</Type /Page
    /Contents 5 0 R
  >>

endobj
5 0 obj
  << /Length 3 0 R >>
  stream
   /P <</MCID 1>> BDC
      BT T* (The most popular sports are:) Tj ET EMC
   /Lbl <</MCID 11>> BDC
      BT T* (1. ) Tj ET EMC
   /LBody <</MCID 12>> /BDC
      BT (Snow-shoeing ) Tj ET EMC
   /Lbl <</MCID 21>> BDC
      BT T* (2. ) Tj ET EMC
   /LBody <</MCID 22>> /BDC
      BT (Ice-skating ) Tj ET EMC
   /Lbl <</MCID 31>> BDC
      BT T* (3. ) Tj ET EMC
   /LBody <</MCID 32>> /BDC
      BT (Skiing ) Tj ET EMC
endstream
endobj

101 0 obj                  % Structure element for intro paragraph to list ("The most popular
  << /Type /StructElem
     /S /P
     /P 201 0 R
     /Pg 4 0 R
     /K 1
  >>
endobj

111 0 obj                   % Structure element for first item, list label (Lbl): "1."
  << /Type /StructElem
     /S /Lbl
     /P 211 0 R
     /Pg 4 0 R
     /K 11
  >>
```

```
endobj

112 0 obj
  << /Type /StructElem     % Structure element for first item, list text (LBody): ("Snow-sho
     /S /LBody
     /P 211 0 R
     /Pg 4 0 R
     /K 12
  >>
endobj

... [ objects 121-122 and 131-132, referencing MCIDs 21-22 and 31-32 are omitted in the inte

201 0 obj
  << /Type /StructElem
     /S /Caption            % Intro paragraph
     /P 400 0 R
     /K [101 0 R]
  >>
endobj

211 0 obj
  << /Type /StructElem
     /S /LI                 % List item for "1. Snow-shoeing"
     /P 400 0 R
     /K [111 0 R 112 0 R]
  >>
endobj

212 0 obj
  << /Type /StructElem
     /S /LI                 % List item for "2. Ice-skating"
     /P 301 0 R
     /K [121 0 R 122 0 R]
  >>
endobj

213 0 obj
  << /Type /StructElem
     /S /LI                 % List item for "3. Skiing"
     /P 301 0 R
     /K [131 0 R 132 0 R]
  >>
endobj

400 0 obj
  << /Type /StructElem
     /S /L                   % Top-level structure element in the list hierarchy
```

```
    /K [201 0 R 211 0 R 212 0 R 213 0 R]
>>
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 14.8.4.3.3 (List Elements) in PDF 1.7 (ISO 32000-1)
- PDF and Accessibility

Related Techniques

- G115: Using semantic elements to mark up structure

Tests

Procedure

1. For a list in a PDF document, verify in one of the following ways:
   - Read the PDF document with a screen reader, listening to hear that list is read correctly when reading the content line-by-line.
   - Use a tool that is capable of showing lists to open the PDF document and view the list to check that it is correctly structured.
   - Inspect the tag tree to verify that the list is structured according to the PDF specification.
   - Use a tool that exposes the document through the accessibility API and verify that the list is correctly structured.

Expected Results

- #1 is true

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF22: Indicating when user input falls outside the required format or values in PDF forms

Applicability

Tagged PDF documents

This technique relates to:

- Success Criterion 3.3.1 (Error Identification)
  - How to Meet 3.3.1 (Error Identification)
  - Understanding Success Criterion 3.3.1 (Error Identification)
- Success Criterion 3.3.3 (Error Suggestion)
  - How to Meet 3.3.3 (Error Suggestion)
  - Understanding Success Criterion 3.3.3 (Error Suggestion)

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF22. Also see PDF Technology Notes.

Description

The objective of this technique is to notify the user when user input to a field that requires a specific, required format (e.g., date fields) is not submitted in that format.

If the required format is not used, an alert dialog describes the nature of the error in text. This may be accomplished through scripting created by the author (see, for example, SCR18: Providing client-side validation and alert). User agents, such as Adobe LiveCycle can provide automatic alerts (as described in the examples below).

> *Note:* Once the user dismisses the alert dialog, it may be helpful if the script positions the keyboard focus on the field where the error occurred, although some users may expect the focus to remain on the last control focused prior to the alert appearing. Authors should exercise care to ensure that any movement of the focus will be expected. For example, if the alert announces an error in a phone number format, positioning the focus on the phone number field when the alert is dismissed can be regarded as helpful and expected. In some cases, however, this may not be possible. If multiple input errors occur on the page, an alternative approach to error notification should be implemented.

Ensuring that users are aware an error has occurred, can determine what is wrong, and can correct it are key to software usability and accessibility. Meeting this objective helps ensure that all users can complete for-based transactions with ease and confidence.

Labels for required formats in form controls

It is also important that users are aware that an error *may* occur. You can incorporate this information in labels; for example, "Date (MM/DD/YYYY)." See *PDF10: Providing labels for interactive form controls in PDF documents*.

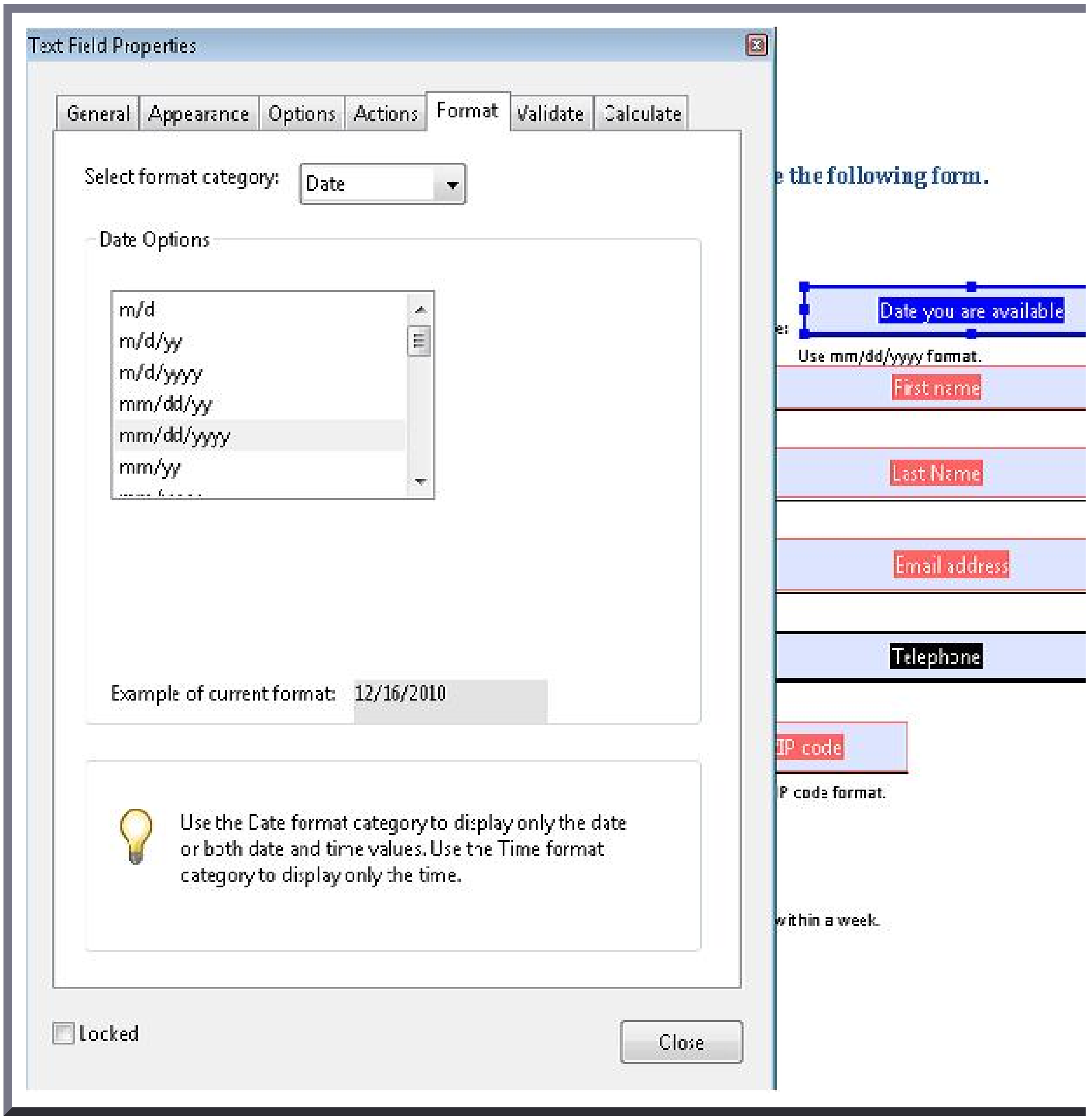
Examples

Example 1: Providing validation for an input field format using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

Many fields -- telephone number, postal code, date -- must have data entered in a specific format or pattern.

1. Access the context menu for the form control that requires a specific format.
2. Select Properties...
3. In the Format tab, select the Format Category (in this case, Date). The Date Options appear.
4. Select the format for this form control (in this case, mm/dd/yyyy).
5. In the General tab, specify "Date (mm/dd/yyyy)" for the Name and Tooltip for the control.



Text Field Properties

General | Appearance | Options | Actions | Format | Validate | Calculate

Select format category: Date

Date Options

m/d
m/d/yy
m/d/yyyy
mm/dd/yy
mm/dd/yyyy
mm/yy

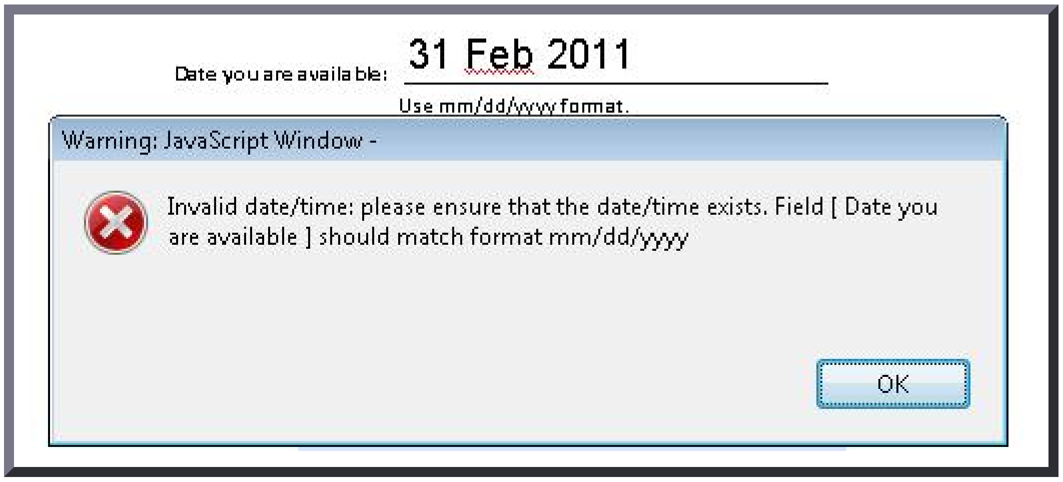
Example of current format: 12/16/2010

Use the Date format category to display only the date or both date and time values. Use the Time format category to display only the time.

Locked

Close

When a user types a recognized date format, it is converted automatically to the specified format. If the date format or value is not recognized, an error alert appears and provides further information, as shown in the image below.



Date you are available: 31 Feb 2011

Use mm/dd/yyyy format.

Warning: JavaScript Window -

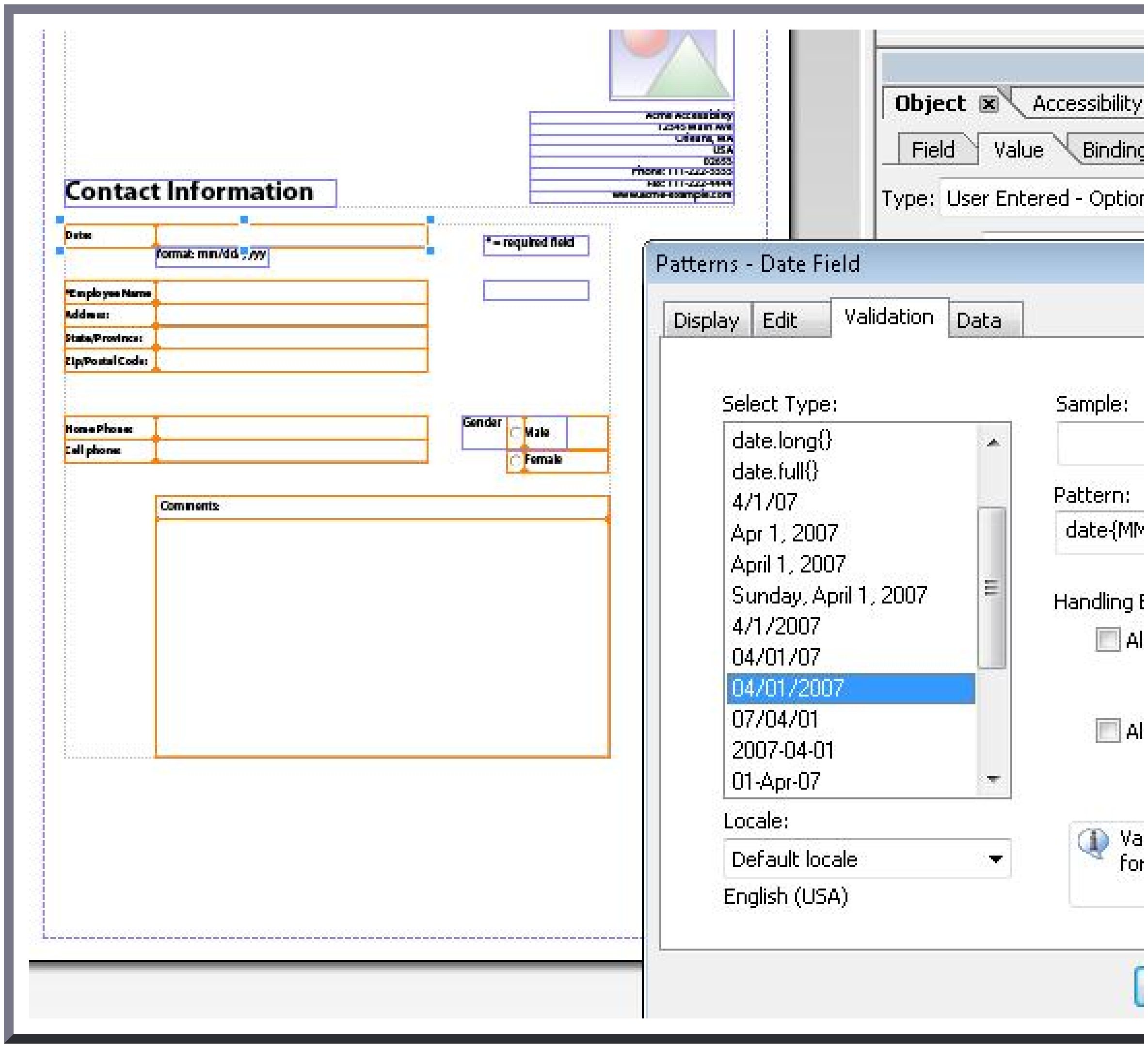
Invalid date/time: please ensure that the date/time exists. Field [Date you are available] should match format mm/dd/yyyy

This example is shown in operation in the working example of Required Fields in Acrobat

Example 2: Providing validation for an input field format using Adobe LiveCycle Designer ES 8.2.1

This example is shown with Adobe LiveCycle Designer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

1. Select the form control that has a required format.
2. In the Object palette, click the Validation Pattern... button.
3. Because this is a date field the Patterns-Date Field dialog appears. Select the pattern or format you want users to enter. Then click OK.



Type: User Entered - Option

Select Type: date.long{} / date.full{} / 4/1/07 / Apr 1, 2007 / April 1, 2007 / Sunday, April 1, 2007 / 4/1/2007 / 04/01/07 / 04/01/2007 (selected) / 07/04/01 / 2007-04-01 / 01-Apr-07

Pattern: date{MM

Locale: Default locale
English (USA)

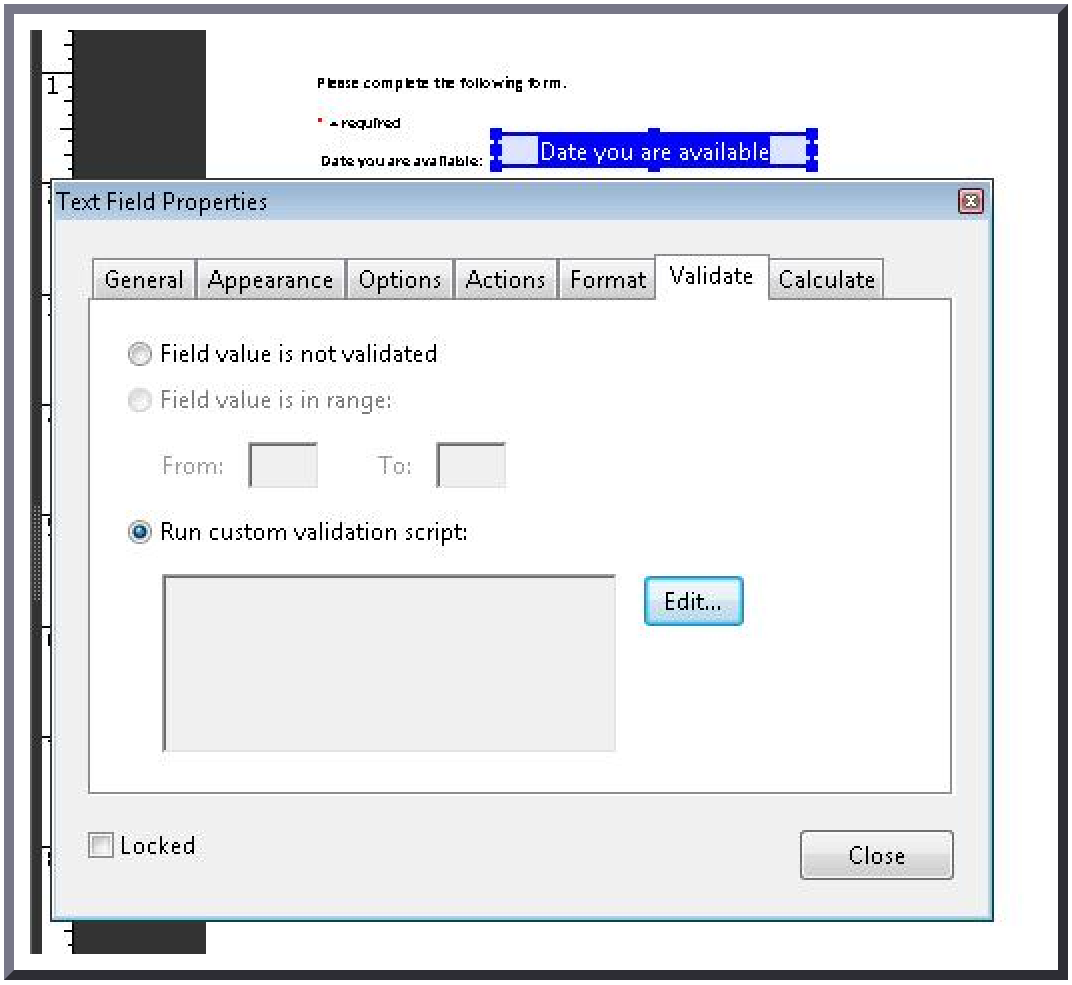
4. In the Object palette, use the Validation Pattern Message box to type a warning message. Be sure to include the required pattern. This message appears when a user tries to submit the form using an invalid date format.

This example is shown in operation in the working example of Required Fields in LiveCycle Designer.

Example 3: Validating a required date format in a PDF form using JavaScript using Adobe Acrobat Pro 9

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

The following JavaScript code illustrates the use of a script to validate form fields, in this case, a date field. To add this script to the form field, open the Text Field Properties dialog, as shown in Example 1, and select Edit in the Validate tab:


Please complete the following form.

* = required

Date you are available: Date you are available

Text Field Properties

General | Appearance | Options | Actions | Format | Validate | Calculate

Field value is not validated

Field value is in range:

From: To:

[✓ Run custom validation script:]

```
// JavaScript code for date mask format MM/DD/YYYY
var re = /^[mdy0-9]{2}\/[mdy0-9]{2}\/[mdy0-9]{4}$/
//Allow blank space in field
if (event.value !="") {
  if (re.test(event.value) == false) {
    app.alert ({
      cTitle: "Incorrect Format",
      cMsg: "Please enter date using mm/dd/yyyy format"
    });
  }
}
```

Resources

Resources are for information purposes only, no endorsement implied.

- JavaScript for Acrobat

Related Techniques

- G89: Providing expected data format and example
- SCR18: Providing client-side validation and alert
- PDF23: Providing interactive form controls in PDF documents
- PDF10: Providing labels for interactive form controls in PDF documents
- PDF5: Indicating required form controls in PDF forms

Tests

Procedure

For each form field that requires specific input, verify that validation information and instructions are provided by applying the following:

1. Check that the format or value that is required is indicated in the form control's label.
2. Use an erroneous format or value and move off the field: make sure that an alert describing the error is provided.

Expected Results

- #1 and #2 are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

✧ ✧ ✧

PDF23: Providing interactive form controls in PDF documents

Applicability

- Tagged PDF documents with forms.
- PDF forms created using Adobe LiveCycle Designer.

This technique relates to:

- Success Criterion 2.1.1 (Keyboard)
  - How to Meet 2.1.1 (Keyboard)
  - Understanding Success Criterion 2.1.1 (Keyboard)
- Success Criterion 2.1.3 (Keyboard (No Exception))
  - How to Meet 2.1.3 (Keyboard (No Exception))
  - Understanding Success Criterion 2.1.3 (Keyboard (No Exception))

User Agent and Assistive Technology Support Notes

See User Agent Support Notes for PDF23. Also see PDF Technology Notes.

Description

The objective of this technique is to ensure that interactive form controls in PDF documents allow keyboard operation. Interactive PDF forms are generally created using a tool for authoring PDF. Form controls are implemented in PDF documents either as described in Section 12.7 (Interactive Forms) of PDF 1.7 (ISO 32000-1) or as described in the Adobe XML Forms Architecture (XFA).

The types of PDF form controls are: text input field, check box, radio button, combo box, list box, and button.

Form controls allow users to interact with a PDF document by filling in information or indicating choices, which can then be submitted for processing. Users who rely on keyboard access must be able to recognize and understand the form fields, make selections, and provide input to complete the forms, and submit the form, just as sighted users can.

Interactive form controls can be provided for forms created by converting a scanned paper form to tagged PDF or by creating a form in an authoring application such as Microsoft Word or Open Office and converting it to tagged PDF.

However, documents created by authoring applications that provide form design features might not fully retain their fillable form fields on conversion to PDF. Complex forms in particular may not have properly converted form fields and labels when tagged in conversion.

Using Adobe Acrobat Pro with forms in converted documents, you can ensure that form fields are keyboard accessible and usable by:

- Opening tagged PDF documents with form fields and creating interactive PDF form elements with the Run Form Fields Recognition tool.
- Modifying fillable form fields, or adding form fields, using Adobe Acrobat Pro or Adobe LiveCycle Designer.

Using Adobe LiveCycle Designer, you can create forms from scratch.

Examples

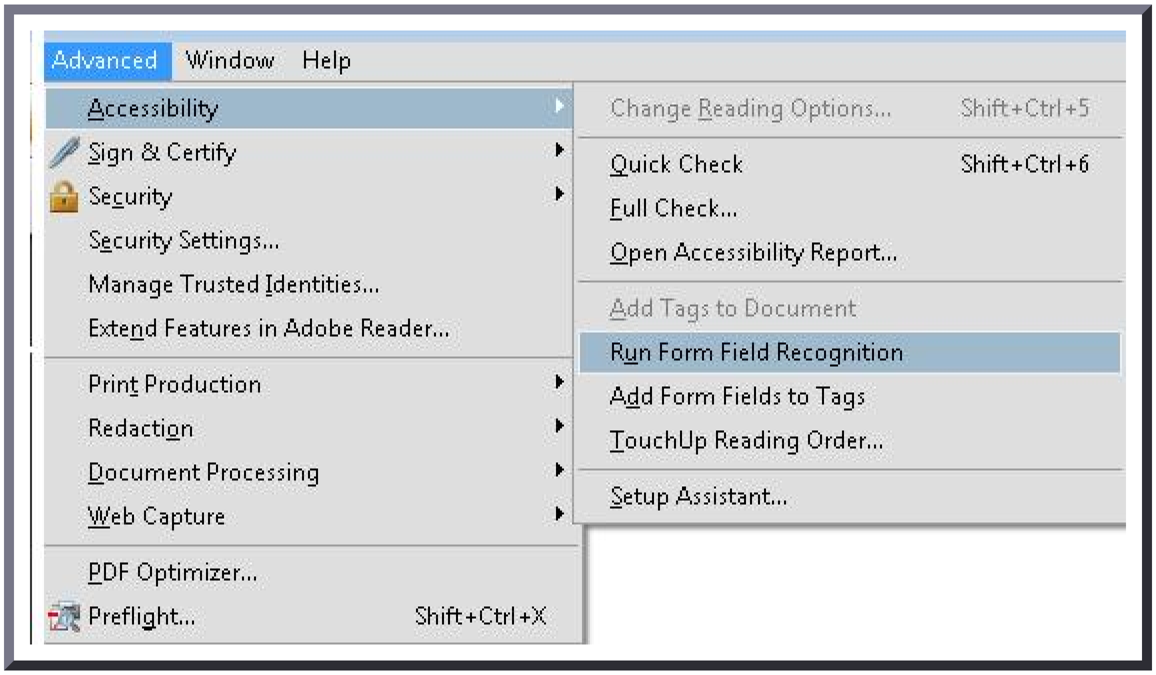
Example 1: Adding interactive controls to existing forms in PDF documents using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

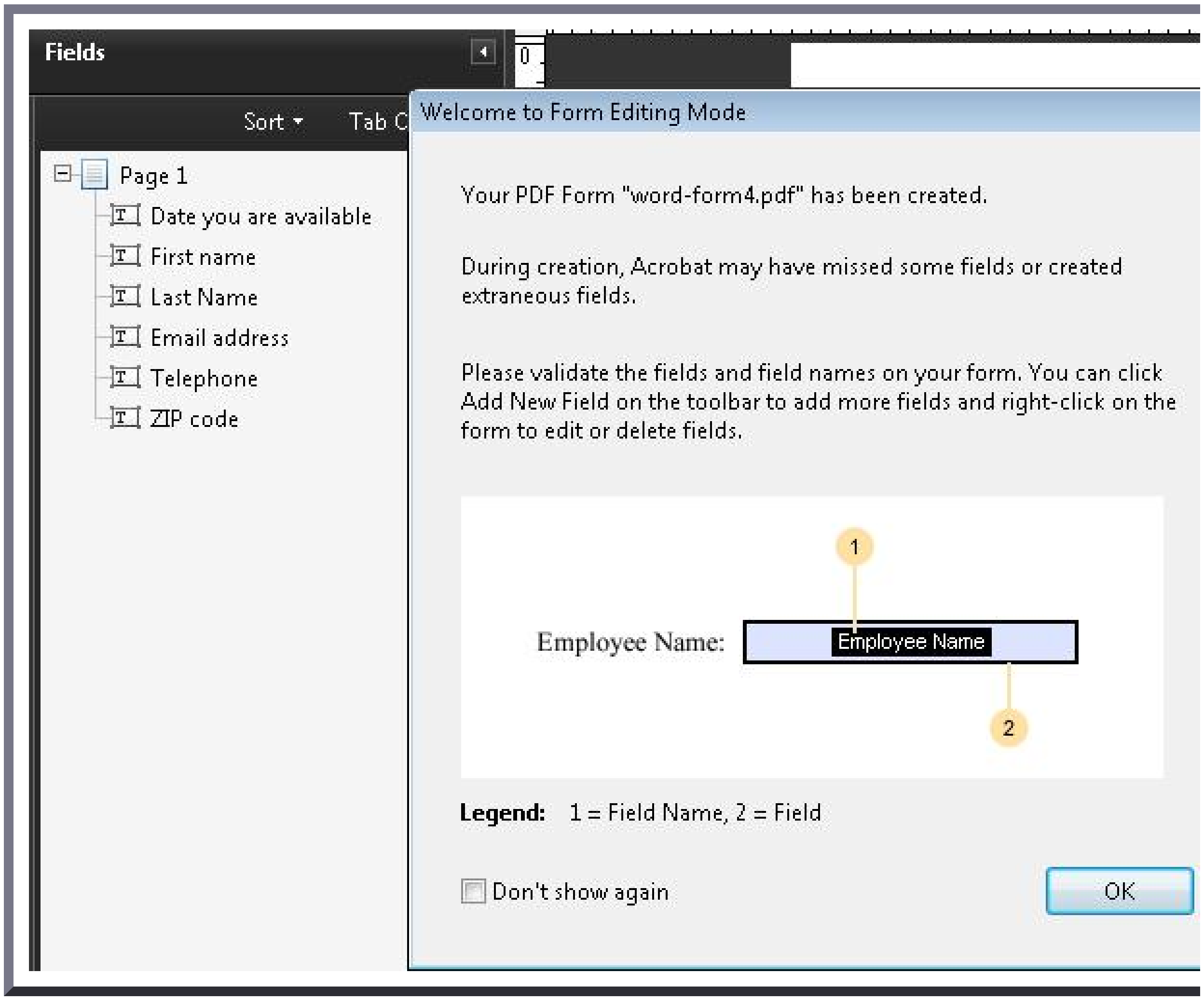
If you have a form in a tagged PDF document (created by scanning a paper form or using an authoring tool to generate tagged PDF), you can use Adobe Acrobat Pro to make the form elements keyboard accessible in the same page locations as the static form.

1. Use Advanced > Accessibility > Run Form Field Recognition to automatically detect form fields and make them fillable.

The following image shows the Run Form Field Recognition tool is selected to detect form fields in a document converted to tagged PDF.



The following image shows the resulting form fields after the Run Form Recognition tool is run.



Your PDF Form "word-form4.pdf" has been created.

During creation, Acrobat may have missed some fields or created extraneous fields.

Please validate the fields and field names on your form. You can click Add New Field on the toolbar to add more fields and right-click on the form to edit or delete fields.

Legend: 1 = Field Name, 2 = Field

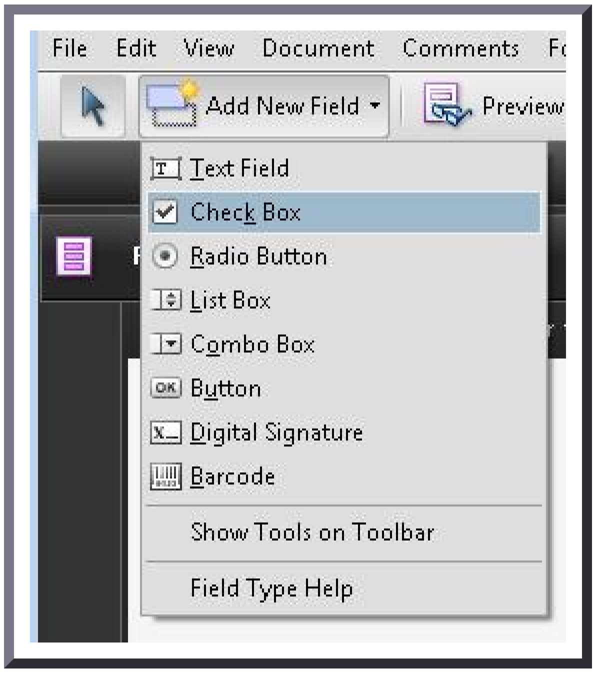
This example is shown in operation in the working example of Interactive Controls in Acrobat.

Example 2: Adding form controls in PDF documents using Adobe Acrobat 9 Pro

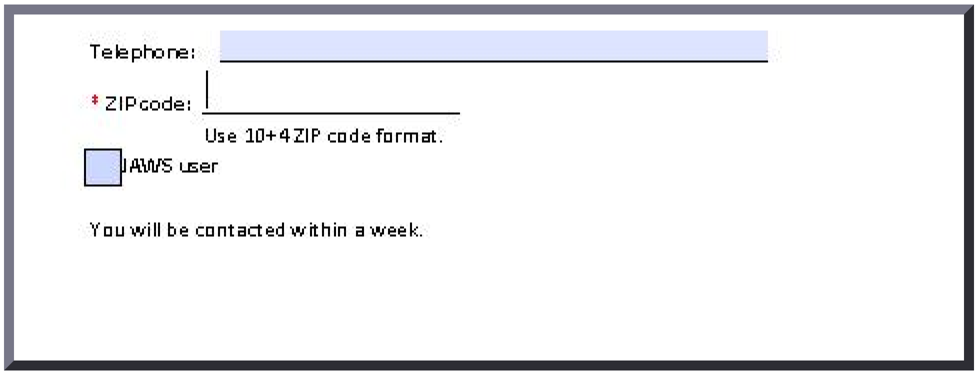
This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

You can add keyboard accessible form controls to your form as follows:

1. Forms > Add or Edit Fields... This puts the form in Form Editing mode.
2. Open the Add New Field menu on the upper left, and select a form field to add. The image below shows the menu of fields.



The following image shows a checkbox added to the form in Example 1.


Telephone:

* ZIPcode:

Use 10+4 ZIP code format.

JAWS user

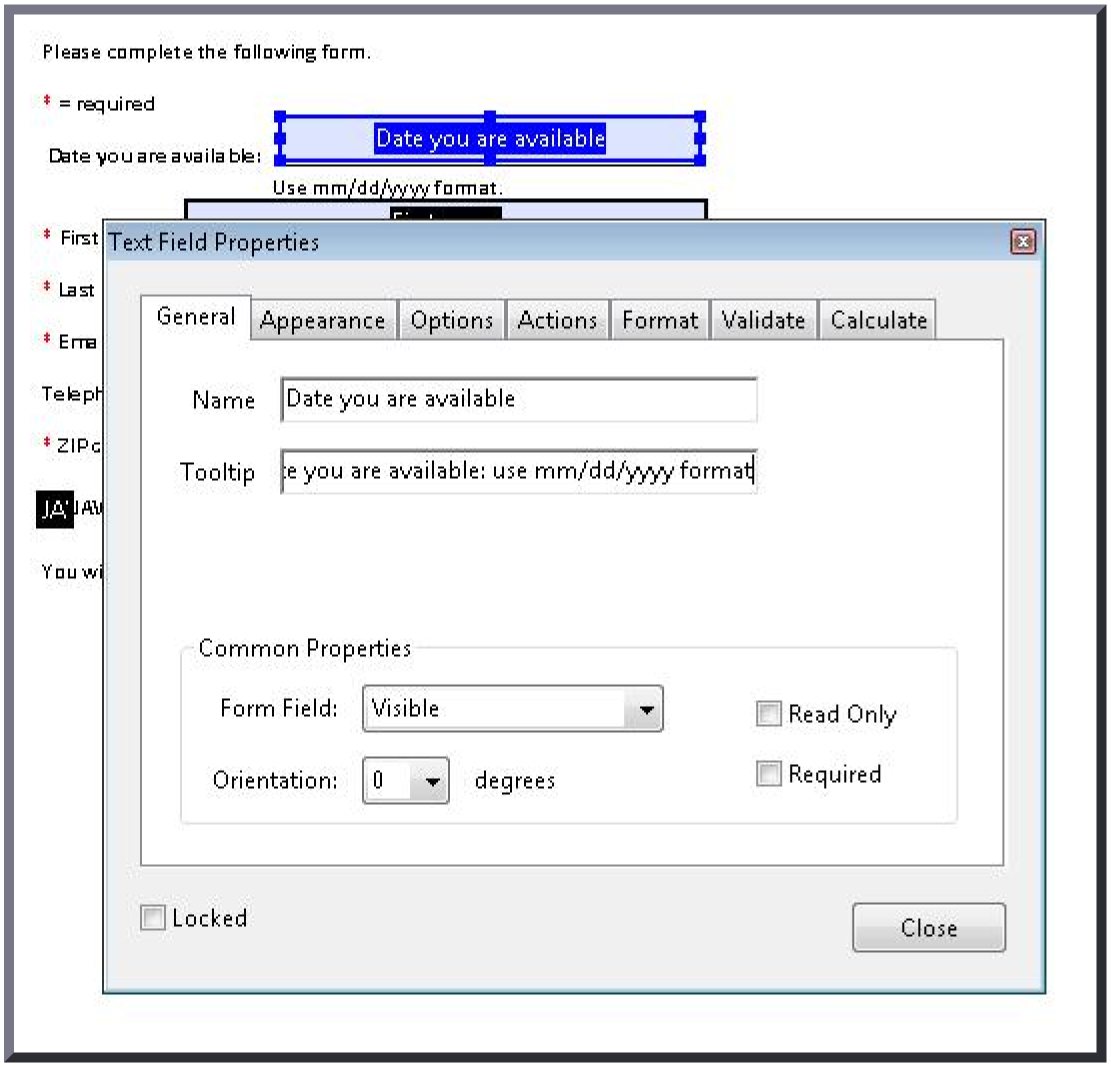
You will be contacted within a week.

This example is shown in operation in the working example of Interactive Controls in LiveCycle Designer.

Example 3: Editing form controls in PDF documents using Adobe Acrobat 9 Pro

This example is shown with Adobe Acrobat Pro. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

To edit fields, select the context menu for the field and select Properties... The properties menu for that form field lets you modify it, as shown in the following image.


Please complete the following form.

* = required

Date you are available: Date you are available

Use mm/dd/yyyy format.

Text Field Properties

General | Appearance | Options | Actions | Format | Validate | Calculate

Name: Date you are available

Tooltip: e you are available: use mm/dd/yyyy format

Common Properties

Form Field: Visible

Orientation: 0 degrees

Read Only

Required

Locked

Close

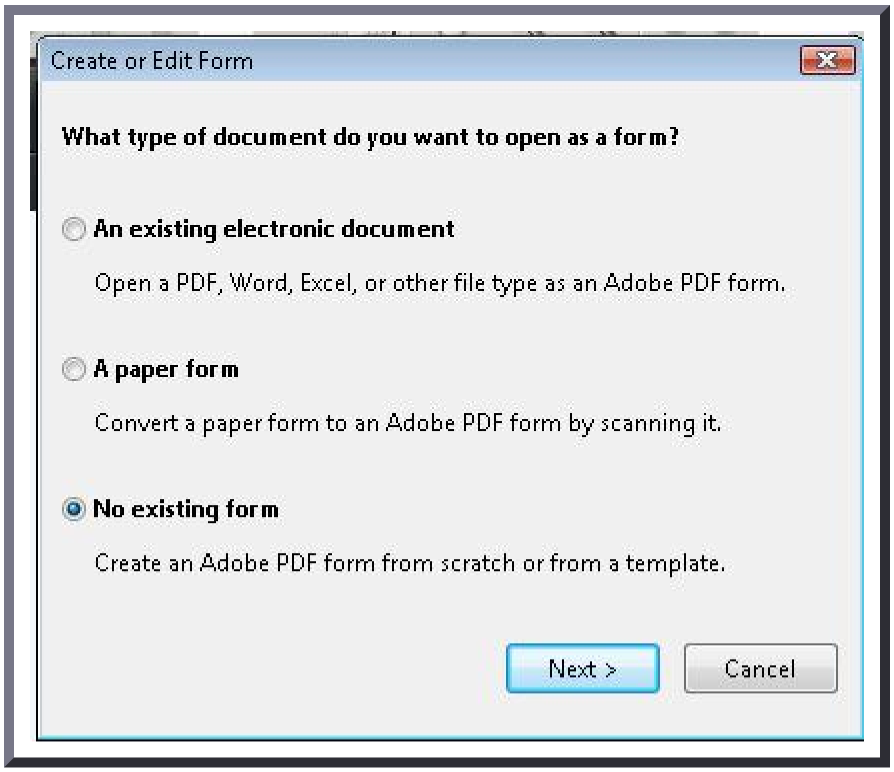
Note: The tooltip is not keyboard accessible but will be screen-reader accessible: see *PDF12: Providing name, role, value information for form fields in PDF documents*.

Example 4: Creating new interactive forms with Adobe LiveCycle Designer ES 8.2.1

This example is shown with Adobe LiveCycle Designer. There are other software tools that perform similar functions. See the list of other software tools in PDF Authoring Tools that Provide Accessibility Support.

You can use Adobe LiveCycle Designer to create new forms. In addition to invoking this standalone tool from the Windows Start menu, you can invoke it in Adobe Acrobat Pro:

1. Forms > Start Form Wizard...
2. Select the No Existing Form radio button, as shown in the following image.



Create or Edit Form

What type of document do you want to open as a form?

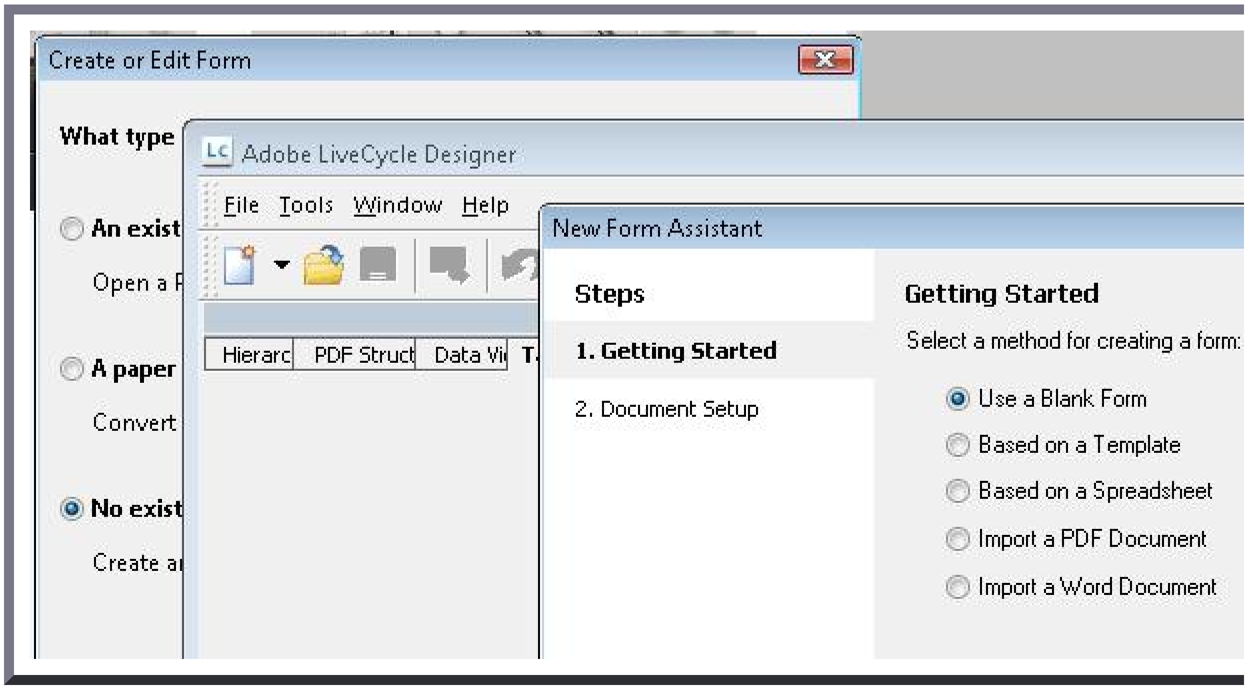
An existing electronic document [✓ No existing form] A paper form

An existing electronic document
Open a PDF, Word, Excel, or other file type as an Adobe PDF form.

A paper form
Convert a paper form to an Adobe PDF form by scanning it.

No existing form
Create an Adobe PDF form from scratch or from a template.

Clicking Next invokes LiveCycle Designer and the first page of the New Form Assistant. as shown in the following image.

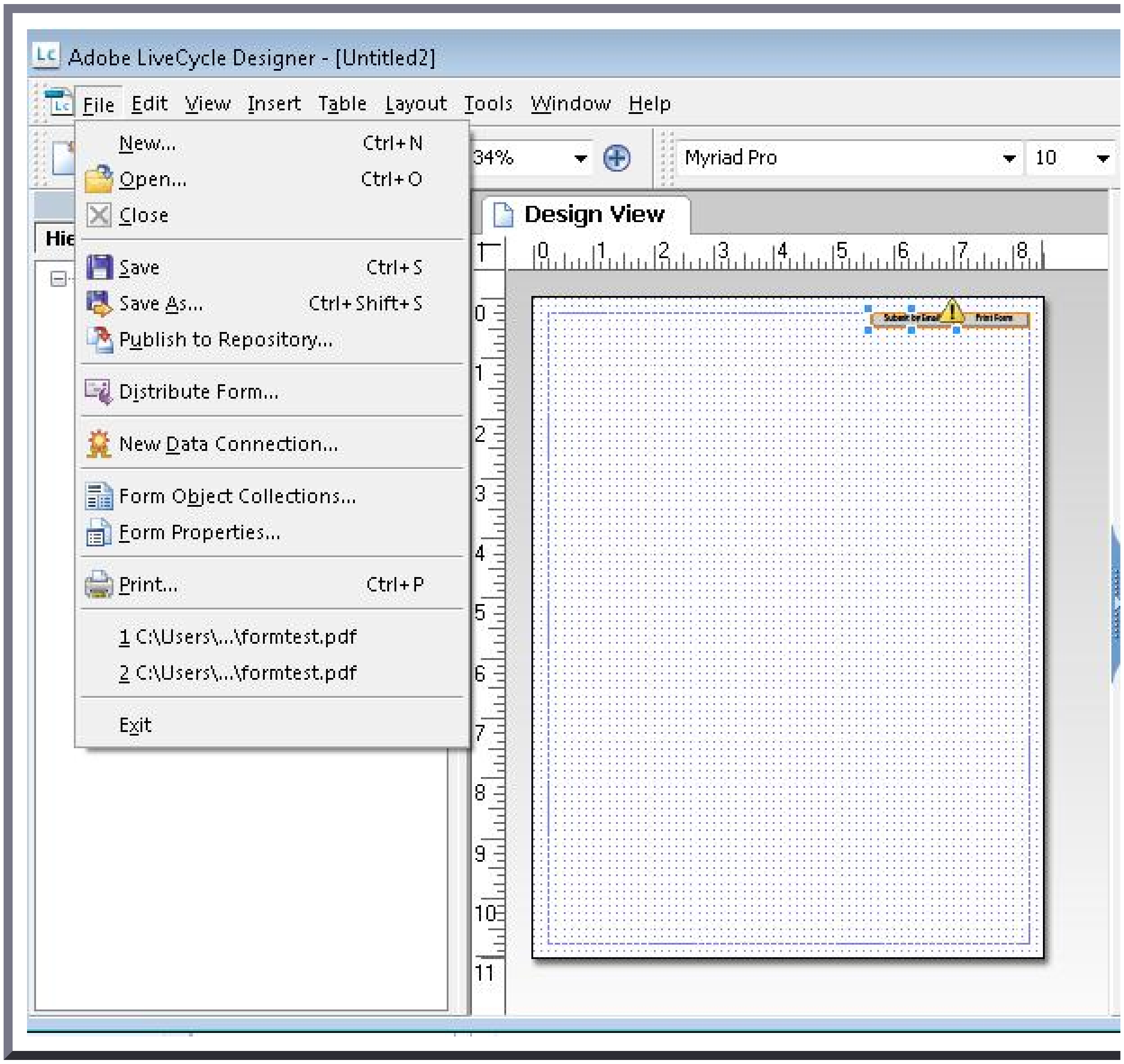


Select a method for creating a form:

- [✓ Use a Blank Form]
- Based on a Template
- Based on a Spreadsheet
- Import a PDF Document
- Import a Word Document

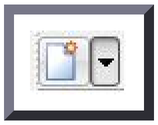
When you invoke LiveCycle Designer from the Windows Start menu, the Form Wizard is available from File > New...

The New Form Assistant creates a blank form. Use the Object Library in the right pane to select form controls.



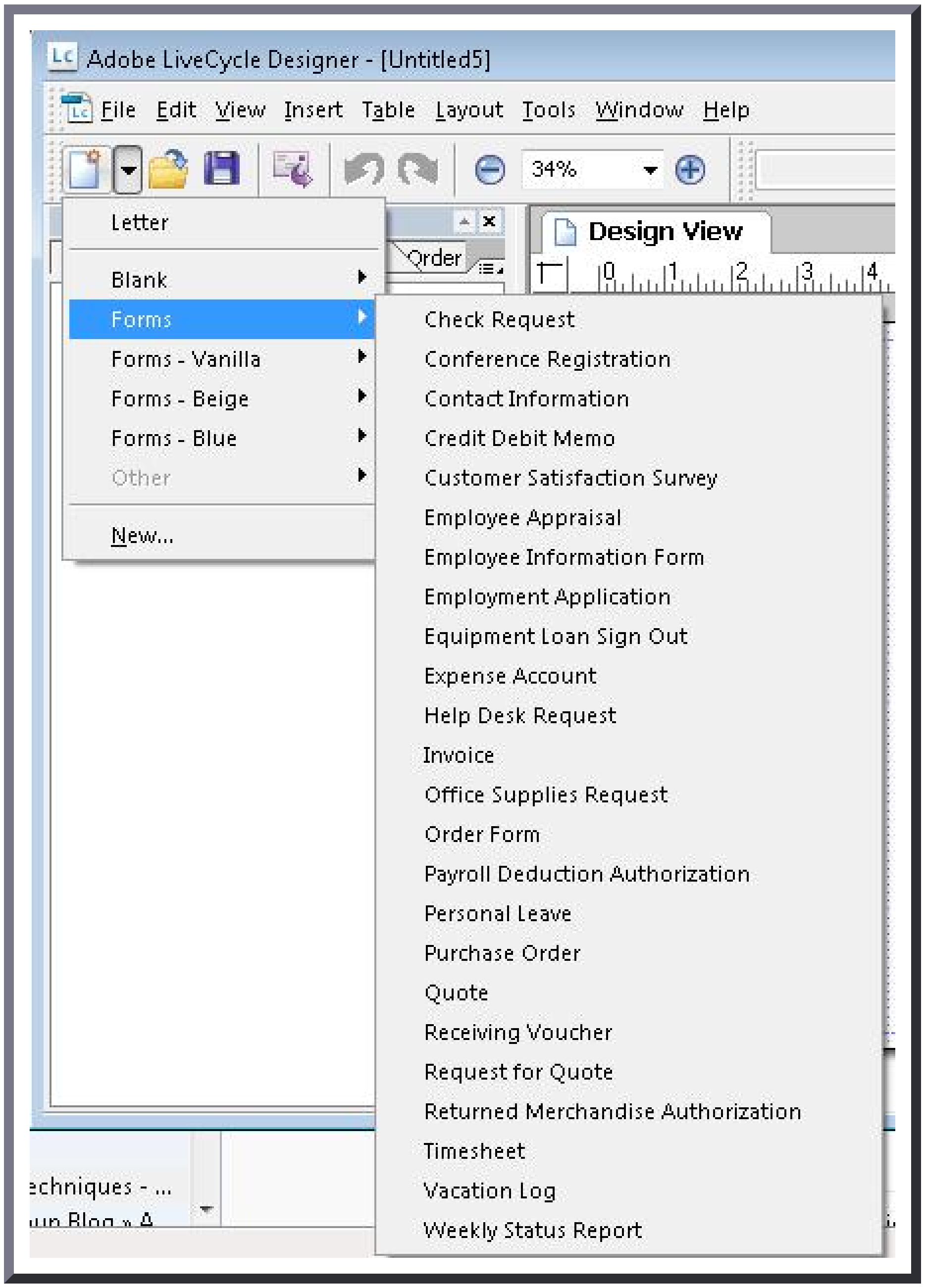
You can also use LiveCycle Designer to create forms based on commonly used forms templates.

1. Invoke the Template Assistant wizard from the New pulldown:

   .

2. Select Forms and then select an appropriate type of form. Then, you can personalize the form by swapping out placeholder text, graphics, form fields, and properties with custom objects that you provide or define.


Adobe LiveCycle Designer - [Untitled5]
File Edit View Insert Table Layout Tools Window Help
34%
Design View
Letter
Blank
Forms
Forms - Vanilla
Forms - Beige
Forms - Blue
Other
New...
Check Request
Conference Registration
Contact Information
Credit Debit Memo
Customer Satisfaction Survey
Employee Appraisal
Employee Information Form
Employment Application
Equipment Loan Sign Out
Expense Account
Help Desk Request
Invoice
Office Supplies Request
Order Form
Payroll Deduction Authorization
Personal Leave
Purchase Order
Quote
Receiving Voucher
Request for Quote
Returned Merchandise Authorization
Timesheet
Vacation Log
Weekly Status Report

Example 5: Adding a text field in a PDF document using the /Tx field type

The following code fragment illustrates code that is typical for a simple text field such as shown in Examples 1 and 2. This is typically accomplished by an authoring tool.

```
<< /AP -dict-
   /DA /Helv  0 Tf 0 g
   /DR -dict-
   /F 0x4
   /FT Tx              % FT key set to Tx for Text Field
   /P -dict-
   /Rect -array-
   /StructParent 0x1
   /Subtype Widget
   /T Date you are available   % Partial field name Date
   /TU Date you are available: use mm/dd/yyyy format % TU tool tip value serves as short des
   /Type Annot
   /V Pat Jones
>>
...
<Start Stream>
 BT
  /P <</MCID 0 >>BDC
  /CS0 cs 0  scn
  /TT0 1 Tf
    -0.001 Tc 0.003 Tw 11.04 0 0 11.04 72 709.56 Tm
    [(P)-6(le)-3(as)10(e)-3( )11(P)-6(rin)2(t)-3( Y)8(o)-7(u)2(r N)4(a)11(m)-6(e)]TJ
  0 Tc 0 Tw 9.533 0 Td
  ( )Tj
  -0.004 Tc 0.004 Tw 0.217 0 Td
  [(\()-5(R)-4(e)5(q)-1(u)-1(i)-3(r)-3(e)-6(d)-1(\))]TJ
 EMC
  /P <</MCID 1 >>BDC
  0 Tc 0 Tw 4.283 0 Td
  [( )-2( )]TJ
   EMC
   /ArtifactSpan <</MCID 2 >>BDC
   0.002 Tc -0.002 Tw 0.456 0 Td
  [(__)11(___)11(___)11(___)11(___)11(_)11(____)11(___)11(___)11(__)]TJ
  0 Tc 0 Tw 13.391 0 Td
  ( )Tj
  EMC
 ET
<End Stream>
```

Resources

Resources are for information purposes only, no endorsement implied.

- Section 12.7 (Interactive Forms) in PDF 1.7 (ISO 32000-1)
- Adobe XML Forms Architecture (XFA)

Related Techniques

- G202: Ensuring keyboard control for all functionality
- PDF3: Ensuring correct tab and reading order in PDF documents
- PDF12: Providing name, role, value information for form fields in PDF documents
- PDF15: Providing submit buttons with the submit-form action in PDF forms

Tests

Procedure

1. For each form control, verify that it is properly implemented by tabbing to each form control and checking that it can be activated or that its value can be changed from the keyboard.

Expected Results

- #1 is true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

CSS1
"Cascading Style Sheets, level 1," B. Bos, H. Wium Lie, eds., W3C Recommendation 17 Dec 1996, revised 11 Jan 1999. Available at http://www.w3.org/TR/REC-CSS1/.

CSS2
"Cascading Style Sheets, level 2," B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds., W3C Recommendation 12 May 1998. Available at http://www.w3.org/TR/CSS2/.

CSS21
"Cascading Style Sheets, level 2 revision 1," B. Bos, T. Çelik, I. Hickson, H. Wium Lie, eds., W3C Candidate Recommendation 25 February 2004. Available at: http://www.w3.org/TR/CSS21/.

CSS3
[CSS 2.1 and CSS 3] Roadmap, CSS WG's table of modules and publication dates.

HTML4
"HTML 4.01 Specification," D. Raggett, A. Le Hors, I. Jacobs, eds., W3C Recommendation 24 December 1999. Available at http://www.w3.org/TR/html401/.

ISO32000
"Document management - Portable document format - Part 1: PDF 1.7", ISO/TC 171/SC 2. ISO. Available at http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=51502. ISO-approved copy available at: http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf.

WCAG20
"Web Content Accessibility Guidelines 2.0," B. Caldwell, M. Cooper, L. Guarino Reid, and G. Vanderheiden, eds., W3C Working Draft 11 December 2007. This W3C Working Draft is available at http://www.w3.org/TR/2007/WD-WCAG20-20071211/. The latest version of WCAG 2.0 is available at http://www.w3.org/TR/WCAG20/.

XHTML1
"XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)," S. Pemberton, et al., W3C Recommendation 26 January 2000, revised 1 August 2002. Available at: http://www.w3.org/TR/xhtml1/.

ACCESSIBILITY.com

Resources > Blog

What Are Accessible Fonts?

ACCESSIBILITY.com

Email Address

Sign Up

LATEST

Top Companies Face Multiple For Accessibility Failures

Understanding Manual and Au Accessibility Testing

Accessible Design and the Inte Things in 2023

How Your Smartphone Can He Have Vision Loss

How Your Smartphone Can He Have Hearing Loss

Search
Within Accessibility Blog

TOPICS

Research and Stats

Digital Accessibility

Website Accessibility

Employment

Physical Accessibility

Education

Technology and Innovation

Legal

Digital Accessibility Laws

By **Cam Waller**
Published November 2, 2021

With text making up so much of the digital content that we consume, keeping it accessible is a crucial step in reducing barriers to information, so typography, typeface, and font selection are all things to take into consideration when planning and producing content. While typography refers to how the text is actually presented on a page or interface, typefaces are the sets of characters used to generate that text—for example, Arial, Times New Roman, and Droid Sans—and each one is made up of a set of fonts. Each of these fonts has a unique combination of characteristics including weight, style, and size. So a bold size 10 Arial would be one font while bold size 20 Arial would be another.

Though the Web Content Accessibility Guidelines (WCAG) don't define specific accessible fonts, the principle for perceivable content states that "information and user interface components must be presentable to users in ways they can perceive," which can serve as the foundation for selecting and using fonts with accessibility in mind.

Choose widely available fonts

While there are several typefaces and fonts available to choose from these days, using the most widely available options is a good practice. This makes it easier to ensure that users will have access to the content in the way you intend to present it and that the experience will be more consistent across different devices and browsers.

Helvetica and Arial are among the fonts that are considered the most common and safest to use. Others such as Verdana are also becoming more widely used. Since browsers will rely on their own default when a font isn't available, in the case that an uncommon or completely custom font is being used, defining fallback options is a great way to help create a more consistent experience.

Opt for fonts that are easy to read

WCAG guideline 1.4 defines how to make content more distinguishable. In relation to text content, this looks at ways of ensuring text is easy to see and that there is a clear separation between the foreground and background.

In most cases, simpler typefaces will be easier to read, and very complex, decorative, or isn't universally true. Sometimes sans serif typefaces may have characters that look very similar, so choosing one with distinctly designed characters will help improve readability.

One challenge that may come up when selecting fonts is the fact that an option that could be easier for some people to perceive may be more difficult for others. For example, the simplicity of a sans serif font may improve the readability of text for a user with a visual impairment while a user with dyslexia may find the characters difficult to tell apart. There are some fonts that have been designed specifically for accessibility, such as Tiresias and OpenDyslexic, but common options such as Arial, Helvetica, Open Sans, and Verdana are also solid choices.

Accessible usage practices

Even when using a typeface that is considered accessible, it's possible to use it in ways that can create barriers for users. To avoid this, using fewer fonts and using them in a way that users expect is preferred. The minimum expected font size is usually 12 points, but 16 points can improve readability for many. It's considered best practice to avoid using bolded, italicized or otherwise stylized fonts to convey meaning, and having the right balance of contrast is important. Though pure white backgrounds and pure black text may seem like an obvious choice the contrast can actually be too strong, so opting for off-white and off-black is often a better choice.

WCAG success criterion 1.4.12 provides guidance on how to meet text spacing requirements, stating that the visibility and functionality of content must not be lost if the user sets line height to 1.5 times the font size, paragraph spacing to 2 times the font size, letter spacing to .12 times the font size and word spacing to .16 times the font size.

When it comes to choosing and using fonts, it may be challenging to meet every individual's needs, but by making mindful decisions, listening to users, and responding to their feedback, creating an experience that is accessible to a large audience is possible.

Vendor Directory

Accessibility.com now offers an impartial listing of digital accessibility vendors. Search for products and services by category, subcategory, or by company name. Check out our new Vendor Directory here.

COMMENTS

First Name *

Last Name

Email *

ACCESSIBILITY.com

Comment *

protected by reCAPTCHA
Privacy - Terms

Submit Comment

RECENT POSTS

December 16, 2021
Conferences and Accessibility: Effective Communication Tools

November 22, 2021
Developing Accessible User Stories

November 17, 2021
Examining the Past, Present, and Future of Assistive Technology

Stay Informed

ACCESSIBILITY.com

Privacy About us Accessibility Advertising Terms & Conditions Contact