Fully Accessible PDF/UA documents. Case study: NOAA fish stock reports

Ross Moore

Macquarie University, Sydney, Australia
ross.moore@mq.edu.au

Abstract

It has frequently been said that PDF files are not accessible, yet much work has been done to develop standards such as PDF/UA to add 'accessibility features' into PDF documents. Using an 80+ page real-world PDF/UA document, built using LaTeX methods, its accessibility is tested using the PAC 2021 validation software, passing all tests with neither violations nor warnings. Yet there is no testing of a significant number of the WCAG recommendations.

To better test we "derive" PDF/UA documents to HTML-5 format and check Accessibility using the AInspector for Firefox browser plug-in; again with neither violations nor warnings, but with numerous tests marked for 'Manual Checking'. In particular, top-level structuring via 'Landmarks' and 'Regions' is confirmed, as well as all hyperlinks having a meaningful and unique 'Accessible Name'. Links, images and other structures have 'Accessible Descriptions', which allow a better understanding of the purpose of the content by a non-visual reader.

With some manual checking of these, helped by an understanding of how the 'Accessible Name' and 'Accessible Description' are constructed, one concludes that the appropriate guidelines are indeed satisfied, thus resulting in a document that is very Accessible. Since the HTML page was derived from the PDF/UA, one concludes that the PDF itself contains all the information to support it being very Accessible also. The only problem is that there may not yet be readily available software that can extract the information in a way that is sufficiently convenient for users with disabilities.

Conclusion: one should not complain that the PDF format is inaccessible; rather, ask for better software that can take advantage of the features that PDF/UA can make available. And similarly demand that publishers prepare PDF/UA documents that are enriched with these features.

Non-readability of a PDF document.

Most PDF documents are built using compression, not just of images, but encryption of entire document segments including textual portions, as in Figure 1. There are many advantages to using compressed PDFs[1]. While such a file may be able to be opened with simple text editor software, and some parts of it may well be readable, most of the information content cannot even be located, let alone extracted. Any attempt to edit it will almost certainly compromise the integrity of the file. This is true for a fully-sighted user, as much as for a non-visual one.

When uncompressed, a PDF file can be seen to be essentially a container for numerous resources known as 'objects', together with an '`xref`' table listing the byte offset from the start of the file to where each object commences. The filesize of the PDFs used for the figures here is roughly 6.8 Mbytes compressed, but ≈34.8 MB uncompressed, containing ≈9600 objects. For

[1] https://www.quora.com/What-are-the-advantages-of-using-compressed-Portable-Document-Format- PDF-files-over-normal-not-compressed-PDFs

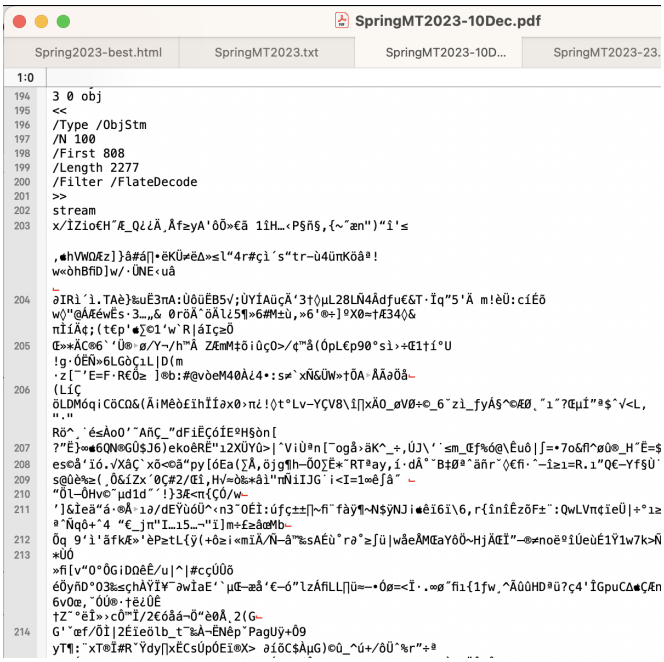


Figure 1. Encryption of portions of a PDF document — intentionally unreadable.

each visual page, there is a 'content stream' containing the graphics instructions to place each word onto that page, often broken into pieces. This is mixed-in with commands for selecting the font, and exact positioning on the page. Other non-visible page elements can be present, especially when tagging is included. This is still quite intractable, as in Figure 2, though some (parts of) words can be identified. The length (in bytes) is also included with such stream objects. If a stream object were to be edited, resulting in a change of length, then not only does the new length need to be recorded, but the byte offsets to every subsequent object would need to be adjusted in the '`xref`' table. This can amount to thousands of edits for just a simple adjustment; clearly this is not a viable way to do it.

Put simply, text-editing software is just not appropriate for handling a PDF file.

Proper text-extraction from a PDF document.

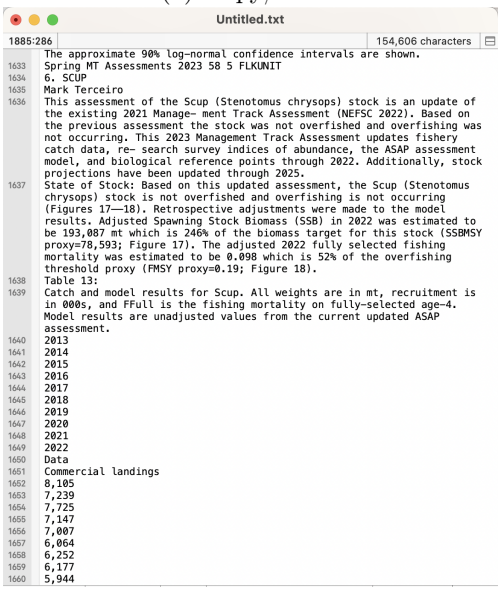
Clearly special software is needed to sift through the complexity of a PDF file's contents in order to build the picture for on-screen display, or to just locate all the textual pieces. One such well-known piece of software is Adobe's free Acrobat Reader[2]. As well as displaying a high-quality visual view, this allows Copy/Paste of the contents; see Figure 3(a). It puts all the word pieces back together, modulo hyphenation, but gives no real idea of the semantics and includes 'Artifact' text such as the running headers and footers including page-numbering.

Acrobat Reader[2] also allows the contents of a PDF to be 'Read Out Loud', as well as saving as Text(Accessible); see Figure 3(b). This latter method is saving the words that would be spoken; which can include extra 'spoken tags' to convey semantics, should the PDF have been constructed to include such a feature. With punctuation not being explicitly spoken, the ';' and ':' characters introduce slight pauses; just a little longer than that of a comma. Notice that 'Artifacts' are excluded, and acronyms have been setup to be spoken letter by letter. With tabular material, row and cell boundaries are clearly identifiable, as well as which cells are meant to be a header for their column or row.

```
6273 0 obj
<<
/Length 40314
>>
stream
0 0 0 rg 0 0 0 RG
0 0 0 rg 0 0 0 RG
1 0 0 1 54 732.719 cm
/Artifact <</Type /Pagination /SubType /Header /Attached [/Top]>> BDC
1 0 0 1 504 0 cm
EMC
0 0 0 rg 0 0 0 RG
1 0 0 1 -504 -24.907 cm
/T <</MCID 0 >> BDC
1 0 0 1 -54 -707.812 cm
BT
/F85 11.9552 Tf 83.265 695.857 Td [(number)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 39.512 0 Td [(out)]TJ/F56 1
Tf( )Tj/F85 11.9552 Tf 18.267 0 Td [(to)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 12.29 0 Td [(the)]TJ/F56 1 Tf( )Tj/
F85 11.9552 Tf 17.598 0 Td [(ar)37(ea)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 24.46 0 Td [(of)]TJ/F56 1 Tf( )Tj/F85
11.9552 Tf 12.289 0 Td [(the)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 17.598 0 Td [(surve)30(y)]TJ/F56 1 Tf( )Tj/F85
11.9552 Tf 33.833 0 Td [(r)37(esulted)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 41.065 0 Td [(in)]TJ/F56 1 Tf( )Tj/F85
11.9552 Tf 12.29 0 Td [(an)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 14.944 0 Td [(estimate)]TJ/F56 1 Tf( )Tj/F85
11.9552 Tf 42.835 0 Td [(of)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 12.29 0 Td [(swept-ar)37(ea)]TJ/F56 1 Tf( )Tj/F85
11.9552 Tf 55.674 0 Td [(ab)20(undance)15(.)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 58.736 0 Td [(At)]TJ/F56 1
Tf( )Tj/F85 11.9552 Tf 13.617 0 Td [(other)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf -427.298 -14.446 Td
[(stations,)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 43.181 0 Td [(a)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 8.967 0 Td
[(tr)15(awl)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 28.058 0 Td [(net)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 17.598 0 Td
[(was)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 21.591 0 Td [(deployed)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 46.146 0 Td
[(to)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 12.29 0 Td [(determine)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 50.797 0 Td
[(r)37(ed)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 18.483 0 Td [(cr)15(ab)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 24.723 0 Td
[(se)20(x)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 18.016 0 Td [(r)15(atios,)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 33.701 0
Td [(weights)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 39.524 0 Td [(and)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 20.921 0 Td
[(length)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 32.876 0 Td [(fr)37(equencies)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf
-416.872 -14.446 Td [(by)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 14.274 0 Td [(se)20(x.)]TJ/F56 1 Tf( )Tj/F85 11.9552
Tf 21.723 0 Td [(Mean)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 30.21 0 Td [(weights)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf
39.523 0 Td [(wer)37(e)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 25.787 0 Td [(used)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf
24.903 0 Td [(to)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 12.29 0 Td [(estimate)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf
42.834 0 Td [(swept-ar)37(ea)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 55.675 0 Td [(biomass.)]TJ
ET
1 0 0 1 392.661 666.965 cm
EMC
1 0 0 1 -309.396 -19.427 cm
/T <</MCID 1 >> BDC
1 0 0 1 -83.265 -647.538 cm
BT
/F85 11.9552 Tf 83.265 647.538 Td [(After)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 26.899 0 Td [(the)]TJ/F56 1
Tf( )Tj/F85 11.9552 Tf 17.597 0 Td [(second)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 36.188 0 Td
[(surve)30(y)55(,)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 36.165 0 Td [(it)]TJ/F56 1 Tf( )Tj/F85 11.9552 Tf 9.635 0
```

Figure 2. Uncompressed portion of a PDF content stream — still intractable.

(a) Copy/Paste



(b) Text(Accessible)

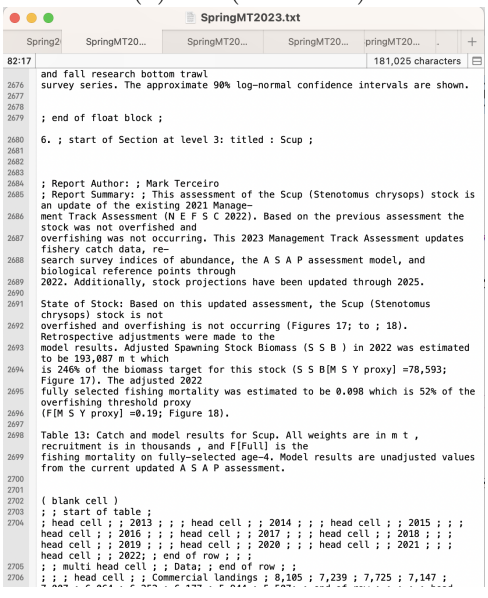


Figure 3. Extraction of text, using Adobe's Acrobat Reader[2] or Acrobat Pro[1]; (a) result from Copy/Paste; (b) result using Save As Text(Accessible).

Adobe's Acrobat Pro[1] is a significant step up from Acrobat Reader[2]. It allows alternative ways to view and access the document contents, and Export into other document formats. Although non-free, Acrobat Pro[1] is used by many academic institutions and can be obtained through site-licensing, perhaps bundled with other useful PDF software. There is a slight difference in its reading and Export to Text(Accessible) as follows. With hyperlinks, Acrobat Reader[2] speaks extra words 'jump to destination . . . ', with the dots being the internal identifier of the destination within the document; the anchor text word(s) are lost. On the other hand, it is precisely these anchor text words that are read (and saved) by Acrobat Pro[1], giving a more natural reading; Figure 3(b) actually shows this result. The Reader words could be useful giving an audible indication of a hyperlink to follow — a precursor of the 'Accessible Name' concept (see below) — but there seems to be no way to choose between these behaviours.

This is a vast improvement but we can do even better, using 'Tagged PDF'.

PDF/UA and Accessibility

As well as capturing large scale structural constructs, such as Parts, Sections with Headings, Lists, Tables, Figures, etc. and active elements such as hyperlinks, buttons and other controls, the 'Tagged PDF' format also supports having attributes for structural items. Being similar to formats such as HTML, SGML and XML, this allows great flexibility in associating layout specifications to content. It also permits conveying semantics by short 'hints' or with extra descriptive text, especially for content where there is implicit meaning in layout choices such as margin-widths, line-spacings or use of italiced or bold-faced text. The WCAG[33] and WAI-ARIA[32] recommendations describe attributes such as alt, role, aria-label, aria-roledescription used to provide such hints to Assistive Technology, as well as having aria-details, aria-labelledby and aria-describedby to reference parts of the document where descriptions may be found.

The published PDF/UA-1 standard has explicit rules about which structural elements may be parents or siblings of other elements, thereby covering the main requirements of Accessibility. However it is rather weak when it comes to specifying when aria-* attributes should be used to better convey meaning, when it is not so easily deduced from context. PDF/UA-2, to be released early in 2024 [24] by the PDF/UA Technical Working Group[26], has some reference to ARIA[32], but not to the extent exhibited in the documents studied in a later section.

To date there is no validation software[2] for PDF/UA documents that checks whether such attributes are present, thus whether the document can truly be regarded as being Accessible for WCAG 2.0, WCAG 2.1, or WCAG 2.2 [34], as is required to satisfy various Government document guidelines [4, 6, 9, 29].

On the other hand, the requirements of PDF/UA-1 are such that a document can be 'derived to HTML'[7] preserving the main document structures and tagging all of the visible content. One means of doing this is by using the online ngPDF[15] by Dual Lab[8] powered by the iText PDF library[11]. With ngPDF any applicable attributes specified for structure items are mapped to attributes on the resulting HTML tags. Where the resulting file is not strictly valid for HTML-5, incorrect patterns can be fixed with multi-line sed (Stream Editor) commands which reorganise the tagging without creating or destroying any content. HTML documents can be tested for WCAG 2.1 [34] accessibility and conformance to ARIA recommendations [32] using the AInspector for Firefox[5] plugin based upon the OpenA11y library[18].

[2] Although PAC 2021[20] and PAC 2024[21] are the best available for PDF/UA-1 and make claims for WCAG compliance, there are many WCAG/ARIA recommendations that are not checked, as we shall see.

Links to two real-world PDF/UA-1 documents, and the HTML derived and corrected to be fully HTML-5 conformant, can be found in the bibliography, for items FallMT2022[10] and SpringMT2023[30].

We discuss these 'fish stock review' documents further in a later section.

'Accessible Name' and 'Accessible Description'

These concepts, described in detail in [35], are like street signs and brief explanations in a city guide or theme park map. The 'Accessible Name' gives an indication of what is found down the street, while the 'Accessible Description' provides more detail. Their purpose in an electronic document is to make it easier to decide whether to go that way (by following a hyperlink, say) or continue reading at the current location. Assistive Technology (AT) is expected to be able to present the Name and/or Description, so that a choice can be made without losing 'focus'.

Most simply, for the 'Accessible Name' of a structural element (or HTML tag) one uses either the value of an aria-label attribute, or the document text indicated by an aria-labelledby reference, with the latter preferred. For the 'Accessible Description' it is aria-details and aria-describedby references that are followed, if available. Availability of references is highly dependent upon the use of 'Structure Destinations' within the PDF, whether for PDF 1.7 or PDF 2.0[22].

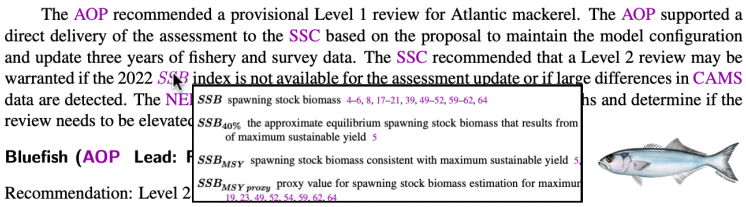


Figure 4. A view of a small portion of the target location "pops up" when the mouse hovers over a link anchor. Move the mouse slightly to dismiss the popup.

In a fully visual context this idea is well displayed as in Figure 4, where in PDF browser software under MacOS, a small popup window shows the contents found at the destination of a hyperlink. There the expansion and meaning of an Acronym is shown without jumping to the Glossary entry itself. Using AT, the 'Accessible Description' would include the expansion 'spawning stock biomass' and there could be an audible cue, perhaps speaking the 'Accessible Name', here the string `Glossary:SSB`. If desired, a button press would deliver the Description; but perhaps not used if the Name is meaningful or familiar as to not require a repeat. Figure 5 shows a representation of the internal PDF objects that encode the link information.

This would be especially relevant when navigating via `taborder` (i.e., using just the `tab` key) in either a PDF or HTML document; especially with a Braille display, where the surrounding context is extremely limited. The text used could be located anywhere within the document, not necessarily at the destination of a hyperlink; though usually there would be some relevant structure involved, such as a caption on an image, chart or photograph. There are WCAG recommendations concerning use of 'Accessible Name' and/or 'Accessible Description', under categories 1.1.1, 1.3.1, 2.4.4, 2.4.6, 3.2.4, 3.3.2 and 4.1.2; related also to headings and landmarks, as well as links, list items, and captions for images, figures and tables.

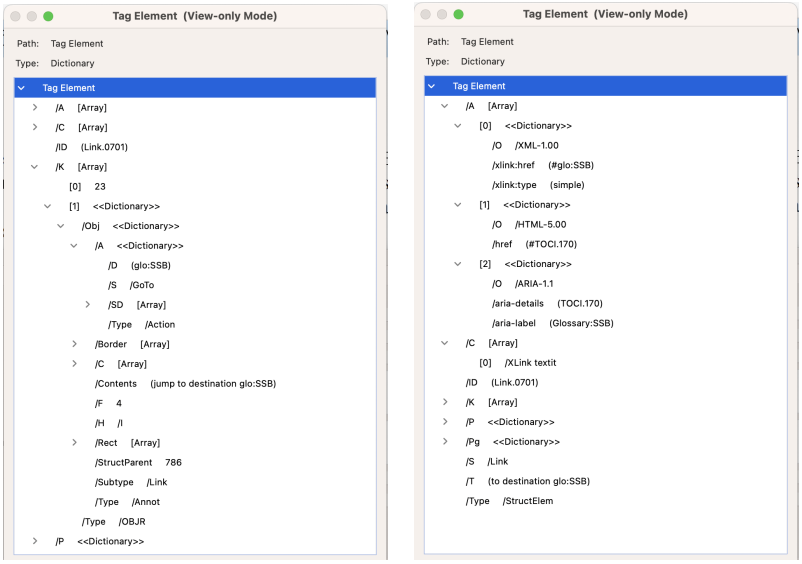


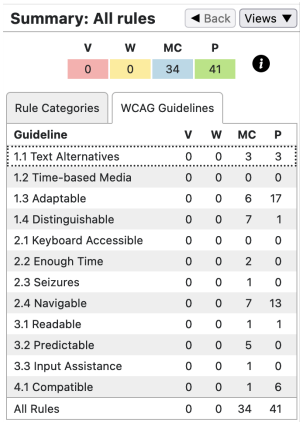
Figure 5. Link structure showing (on left) the `GoTo` action and target destination `glo:SSB`; and (on right) the attributes for Export to XML and HTML, with extra ARIA attributes used to create the 'Accessible Name' and 'Accessible Description' for the Acronym link of Figure 4.

Case study: Fish stock reviews

As an aside from Accessibility, the FallMT2022[10] and SpringMT2023[30] example documents are Technical Memoranda prepared for the NEFSC[14], containing Reports and Peer- and Oversight-reviews of the state of fish stocks in the Atlantic Ocean, off the northeast coast of the U.S.A. Data is collected from fishing vessels at sea or when unloading at a port. This is saved in database files and used with statistical modelling to confirm previously-set parameters, and make predictions for expected catches in future years. Research analysts upload statistics and textual descriptions. At least twice yearly Daniel Hennen (NEFSC[14]) extracts reports, using R[28] programs to create LaTeX source files for tables and figures and to present the textual materials for each fish stock under review. The LaTeX-based structure for these documents was developed by Daniel and Prof. Thomas Price (emeritus, University of Akron). In all there are up to a dozen input files for each fish stock, as well as many other information files that are read during the LaTeX preamble, defining macros for later use. NOAA[16], a U.S. government agency, must meet Accessibility requirements [4, 29]. Due to previous work that Ross has done on 'Tagged PDF' using LaTeX[13], he was invited to join the team to work at augmenting the high-quality visual view with Accessibility enhancements to satisfy all WCAG requirements for this kind of subject matter.

That LaTeX was used to create the PDF/UA documents is not the main issue here. Rather it is the enhancements for Accessibility that can be included through aria-* attributes that allow many WCAG/ARIA [32] recommendations to be met. These can be examined in the HTML version using the AInspector[5] when using the Firefox browser, and with other plug-ins for other commonly-used browsers. The latest, AInspector v3.0 tests 120 separate rules of which 51 have no applicable tag to test within the HTML version of the SpringMT2023[30] example. This leaves significantly more applicable Accessibility tests than are listed in the Matterhorn Protocol[12], or are tested for within PDF/UA documents by veraPDF[31] or PAC 2021[20] or

(a) AInspector summary



(b) PAC 2021, WCAG panel summary

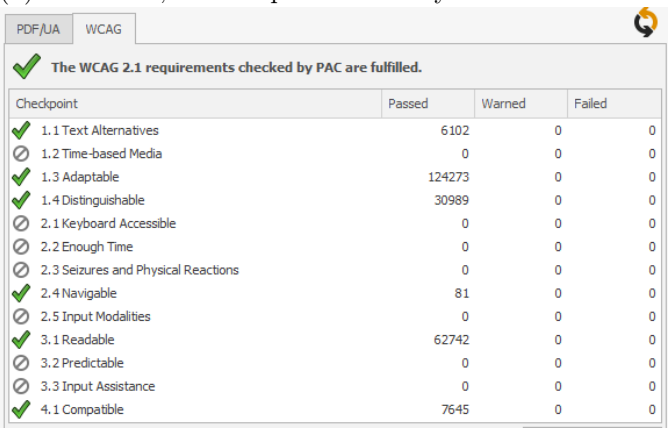


Figure 6. Validation summaries of conformance with applicable WCAG recommendations.

the recently released PAC 2024[21]. With 0 violations and 0 warnings, 41 of the remaining tests are algorithmically checked to be Passing, leaving 34 where a manual check (MC) is advised to determine whether a recommendation is actually satisfied; see Figure 6(a) for a summary.

Many of these remaining rules are indeed automatically satisfied, due to the nature of the PDF/UA format and the HTML document derived from it. The website for SpringMT2023 (see [30] for the link address) gives more details on the specific tests, explaining what needs to be checked or a justification for why a Pass can be presumed. E.g., for rule Navigation 3: 'Consistent ordering of H1 and H2 labels' (ARIA Success Condition 3.2.3), there is an equivalent test in PDF/UA. With a valid PDF/UA there is no manual checking needed. Similarly for rule Link 3: 'Target focus should be in content window', which is part of ARIA Success Condition 3.2.1. As there is just a single window containing the complete HTML site, this is satisfied for all internal hyperlinks. External hyperlinks would have been confirmed when the link target was harvested for use in the PDF document. Hence all 1500+ instances are effectively confirmed. And with rule Table 1: 'Data cells must have row/column headers', part of ARIA Success Condition 1.3.1, we have Pass for 484 table cells. The remaining 28 cells can be easily confirmed to be blank, for formatting purposes only, which is what the test requires.

On the other hand, rule Image 1: 'Alt text must summarize purpose', part of ARIA Success Condition 1.1.1, requires checking that the Alt text for each Figure structure is applicable to the actual image. This should be part of the normal proof-reading and editorial checking. Other tests have confirmations that similarly should be part of normal checking; nevertheless it could be useful to have the reminder.

With essentially all rules checked, using the considerations detailed in the SpringMT2023 website (see the link for [30]), one concludes a very high degree of WCAG conformance for the HTML document. Now since that HTML was 'derived' from a PDF, one realises that the same information that enhances the HTML file's conformance is present also within the PDF; so it too should be considered to have a high degree of WCAG conformance. There is a difficulty with the PDF however, in that there may not be adequate software, easily available, to take advantage of that information. In summary we have the following:

> It is not the PDF, or PDF/UA, format that should be considered as not being Accessible; rather, documents can be created supporting a high degree of Accessibility. The issue is that *software able to take full advantage of this Accessibility is not yet readily available.*

Direct validation in the PDF

Figure 6(b) shows a summary of the validation results from PAC 2021[20] working directly on the PDF/UA document, as organised by WCAG 2.1[34] success criteria. Showing no violations, nor even any warnings, one could easily presume that the document is fully compliant with all WCAG recommendations. However that is over-simplifying, somewhat.

In Figure 6(b) we see that 6 out of 13 categories are indicated as Passing, having run thousands of individual checks on structure elements and associated content. However, there are 7 categories with 0 tests recorded. Possible explanations for having 0 tests can be one of the below, or a combination thereof.

(a) There are no appropriate tests implemented; or

(b) no structure or content in the PDF is appropriate for what the tests need to check.

The WCAG/ARIA[32] recommendations are about ways to make certain kinds of information content and structures more accessible to users/readers having various kinds of disability. Figure 6(a) shows that AInspector[5] has run many tests in those categories, which include some for which the 'Accessible Name' and/or 'Accessible Description' are relevant. As the HTML was derived from the PDF, it is inconceivable that (b) alone could be the correct explanation.

Also it is remarkable that only 81 tests were run in Category '2.4 Navigable'. This includes sub-categories 2.4.4 and 2.4.6 which require looking at the 'Accessible Name' for the 1500+ internal hyperlinks. As well as requesting manual checks that the algorithmically constructed 'Accessible Name' is meaningful, and describes the purpose of the link, there is the requirement that links having the same 'Accessible Name' jump to the same target. The total number of tests run should certainly be in the thousands, well above 81 as reported. So one can only conclude that the 'Accessible Name' concept has not been addressed by PAC software.

Furthermore, within the 6 categories tested by PAC[20], AInspector[5] finds 25 applicable tests that require manual checking. Not all of these can be due to the change of format from PDF into HTML. As well as 2.4.4 and 2.4.6 covered in the previous paragraph, there are tests in subcategories 1.1.1, 1.3.1 and 4.1.2 which are about having extra hints via aria-* attributes.

Recommendations

The 'Tagged PDF' document format already has the capability for production of PDF/UA documents that support WCAG Accessibility criteria, to a very high level of Accessibility. However, there is currently a lack of software that can take full advantage of this, to give users with disabilities a satisfying reading experience.

Thus for better Accessibility (i.e., satisfying more WCAG/ARIA Success Criteria), we make recommendations as follows.

1. Software producing PDF/UA documents should include 'hints' by populating the aria-* attributes used in creating a meaningful, descriptive 'Accessible Name' and 'Accessible Description', with all structure elements where this can reasonably be helpful.
2. PDF consuming Assistive Technology (AT) applications should be written to look for and act upon the presence of aria-* attributes, to provide a better reading experience.
3. Future updates of the PDF/UA standard should describe the purpose and usage of aria-* attributes, along with recommendations for their support.

Acknowledgements

We thank others who have been either directly involved in doing this work, or indirectly via discussions concerning tagging and/or Accessibility. Firstly Daniel Hennen, an Operations Research analyst at NEFSC[14], and Thomas Price, emeritus at University of Akron for inviting me to work on the layout and accessibility aspects of these Technical Memoranda. Next there is Boris Doubrov from Dual Lab[8] and head of the LaTeX Project LWG[27], and Roman Toda from the 'Deriving HTML from PDF' TWG[7]. We wish to acknowledge many discussions (email and online) with other members of the LaTeX Project LWG[27], of which Ross is a member; namely Chris Rowley, Frank Mittelbach, David Carlisle, Ulrike Fischer and others. Some thanks also go to Jon Gunderson of University of Illinois Accessible IT group, a developer of the OpenA11y Evaluation Library[18] and AInspector[5], for sharing his latest versions for beta-testing using the HTML derivations, prior to the release of version 3.0, now renamed as AInspector for Firefox. More recently, we thank Thomas Schempp from axes4 GmbH Zürich, for help with interpreting results from PAC 2021[20] resulting in minor fixes and allowing pre-release testing of PAC 2024[21]. Chris, Dan and Tom also for comments on details in this paper.

Software tools and techniques, organisations, websites

[1] Acrobat Pro, Adobe Inc.; non-free software for viewing, reading, printing, editing and manipulation of PDF files. https://www.adobe.com/acrobat.html

[2] Acrobat Reader, Adobe Inc.; free software for viewing, reading, printing PDF files. https://get.adobe.com/reader/

[3] U.S. Department of Justice, Civil Rights Division: Americans with Disabilities Act of 1990, as amended. https://www.ada.gov/law-and-regs/ada/

[4] U.S. Department of Justice, Civil Rights Division: Guidance on Web Accessibility and the ADA[3], March 2022. https://www.ada.gov/resources/web-guidance/

[5] AInspector for Firefox. University of Illinois Accessible IT group. Uses the OpenA11y Evaluation Library[18]. https://ainspector.disability.illinois.edu.
GitHub: https://github.com/opena11y/ainspector-for-firefox

[6] BS 8878:2010 "Web accessibility. Code of practice." British Standards Institute (BSI), December 2010. https://knowledge.bsigroup.com/products/web-accessibility-code-of-practice?version=standard. Wikipedia: https://en.wikipedia.org/wiki/BS_8878

[7] Deriving HTML from PDF, 2019. A usage specification for tagged ISO 32000-2 files. Publ. by PDF Association[23, 25]. https://pdfa.org/resource/deriving-html-from-pdf/

[8] Dual Lab, software developers providing product development services in multiple domains. https://duallab.com/

[9] European accessibility act: Accessibility standardisation. https://ec.europa.eu/social/main.jsp?catId=1485&langId=en

[10] FallMT-2022: Management Track Assessments Fall 2022. Available in PDF/UA and WCAG 2.1 accessible HTML-5 versions. http://science.mq.edu.au/~ross/TaggedPDF/FallMT2022/

[11] Apryse: iText PDF library: toolkit offering well-documented and versatile PDF engines, written in Java and .NET). https://itextpdf.com

[12] Matterhorn Protocol 1.1. PDF/UA Conformance Testing Model. PDF/UA Technical Working Group[26]. https://pdfa.org/resource/the-matterhorn-protocol/

[13] Moore, Ross: Examples of Tagged PDF documents built using LaTeX. http://science.mq.edu.au/~ross/TaggedPDF/index.html. last updated December 2023.

[14] Northeast Fisheries Science Center. Office of NOAA Fisheries[17]. https://www.fisheries.noaa.gov/contact/northeast-fisheries-science-center-0

[15] ngPDF, by Dual Lab[8], using the iText PDF library[11];
online service at https://ngpdf.duallab.com. Requires validation with veraPDF[31].

[16] NOAA. National Oceanic and Atmospheric Administration; U.S. Department of Commerce. https://www.noaa.gov

[17] NOAA Fisheries. National Oceanic and Atmospheric Administration. https://www.fisheries.noaa.gov. Science & Data: https://www.fisheries.noaa.gov/science-and-data.
Research: https://www.fisheries.noaa.gov/resources/peer-reviewed-research.

[18] OpenA11y Evaluation Library, University of Illinois Accessible IT group. https://accessibleit.disability.illinois.edu/tools/ainspector-wcag/evaluation-library/.
GitHub: https://github.com/opena11y

[19] Open Preservation Foundation: global not-for-profit membership organisation working to advance shared standards and solutions for the long-term preservation of digital content.
https://openpreservation.org

[20] PAC 2021: PDF Accessibility Checker; PDF/UA Foundation. PDF/UA and WCAG[33] validation software, for Windows only. https://pdfua.foundation/en/pdf-accessibility-checker-pac/

[21] PAC 2024: PDF Accessibility Checker; PDF/UA Foundation. PDF/UA and WCAG[33] validation software, for Windows only; developed by axes4 GmbH, Zürich (CH); released December 2023. https://pac.pdf-accessibility.org

[22] PDF 2.0; no-cost access to the latest PDF standard: ISO 32000-2 (PDF 2.0). PDF Association[23]. https://pdfa.org/sponsored-standards/

[23] PDF Association: home of the worldwide PDF technical community. https://pdfa.org

[24] Usage of Tagged PDF in PDF 2.0. PDF Association[23], PDF Reuse & PDF/UA TWG[26]. To appear in 2024.

[25] 'Deriving HTML from PDF' TWG of the PDF Association[23].
https://pdfa.org/community/deriving-html-from-pdf-twg/

[26] PDF/UA Technical Working Group, PDF Association[23]. Drives development of ISO 14289 (PDF/UA) through its association with ISO TC 171 SC 2 WG 9 and publishes resources for developers who need to understand the International Standard for universally accessible PDF.
https://pdfa.org/community/pdf-ua-technical-working-group/

[27] The LaTeX Project Liaison Working Group (LWG), PDF Association[23]. Working on a project to enhance LaTeX to fully and naturally support creation of structured document formats, in particular the "tagged PDF" format as required by accessibility standards such as PDF/UA.
https://pdfa.org/community/latex-project-lwg/

[28] The R Project for Statistical Computing. A programming language and environment for statistical computing and graphics. https://www.r-project.org

[29] U.S. Access Board, Section 508 of the Rehabilitation Act: Information and Communication Technology, Revised 508 Standards and 255 Guidelines, 2018: https://www.access-board.gov/ict/, WCAG 2.0 & PDF/UA: https://www.access-board.gov/files/ict/ict-final-rule.pdf.

[30] SpringMT-2023: Management Track Assessments Spring 2023. Available in PDF/UA and WCAG 2.1 accessible HTML-5 versions. http://science.mq.edu.au/~ross/TaggedPDF/SpringMT2023/

[31] veraPDF, Open source industry supported PDF/A validation maintained by the Open Preservation Foundation[19]. Java application; https://github.com/veraPDF.

[32] Web Accessibility Initiative (WAI). Making the Web Accessible: strategies, standards, and supporting resources to help you make the Web more accessible to people with disabilities.
https://www.w3.org/WAI/

[33] Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation, September 2023. Accessibility Guidelines Working Group[32]. https://www.w3.org/TR/WCAG/

[34] Web Content Accessibility Guidelines 2, Overview. Web Accessibility Initiative[32].
https://www.w3.org/WAI/standards-guidelines/wcag/

[35] Accessible Name and Description Computation 1.1; W3C Recommendation, 18 December 2018. Web Accessibility Initiative[32]. https://www.w3.org/TR/accname-1.1/