To: United States Patent and Trademark Office

August 6, 2023

This is a comment by Carl Oppedahl in response to *DOCX Submission Requirements*, 88 FR 37039 (June 6, 2023).

This comment directs itself to "the quality, utility, and clarity of the information to be collected" in DOCX format. Kindly see three documents attached.

- Oppedahl, Carl, *The Fool's Errand that is DOCX* (December 27, 2022). Available at SSRN: https://ssrn.com/abstract=4346907, attached hereto as Exhibit A, PDF page 3.

- Oppedahl, Carl, *One reason why USPTO's DOCX initiative presents professional liability risks* (June 20, 2023). Available at https://blog.oppedahl.com/?p=9634, attached hereto as Exhibit B, PDF page 39.

- Oppedahl, Carl, *Deficiencies in the "auxiliary PDF" approach for DOCX filing* (June 5, 2023). Available at https://blog.oppedahl.com/?p=9585, attached hereto as Exhibit C, PDF page 43.

Respectfully submitted,

/ s/

Carl Oppedahl
Oppedahl Patent Law Firm LLC

Exhibit A

The Fool's Errand That Is DOCX

Version: 2022-12-27

By: Carl Oppedahl

Table of Contents

Executive summary: Nearly everything that the USPTO has ever said about DOCX is false. To the extent that any word processor format actually satisfies USPTO's stated requirements, it is ODF (OpenDocument text) format. DOCX actually fails to satisfy any of USPTO's stated requirements.

Almost everything about USPTO's DOCX filing process is offensive to patent applicants and practitioners. USPTO's DOCX filing process, if carried out and perpetuated, would expose practitioners to substantial malpractice risks.

There is no “DOCX standard”. There is an “ODF standard”. There is an international industry standard for a type of PDF file that would actually provide everything that the USPTO says it needs.

The USPTO’s “yearlong study” that supposedly showed that PDF was not a good way for filers to file patent applications (from USPTO’s point of view) showed no such thing. In the associated Federal Register notice, the USPTO profoundly mischaracterized the conclusions of the study.

The USPTO has gone out of its way to ignore and mischaracterize comments and suggestions of practitioners about this DOCX problem.

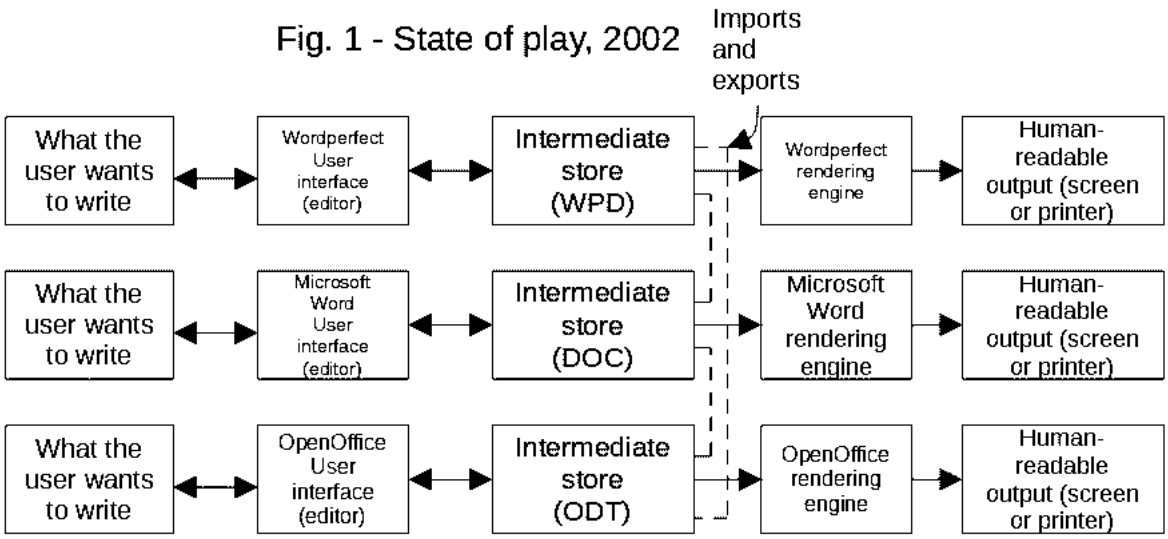
There are two decent ways to fix USPTO’s mess:

- scrap the DOCX program and ask for ODF files instead of DOCX files; or
- scrap the DOCX program and ask for accessible PDF files, along with incentivizing the filer to provide a DOCX file if the USPTO feels it needs the DOCX file.

I am providing this *Fool’s Errand* document to the USPTO (and to the practitioner community) in December of 2022, the timing of which was driven by the USPTO’s doubling down on a January 1, 2023 start date for USPTO’s $400 penalty for failing to comply with USPTO’s requirement that patent applications be filed in Microsoft Word DOCX format. But I remind the USPTO that every individual point made in this *Fool’s Errand* document is a point that I made to USPTO decisionmakers in the past. Nothing in this *Fool’s Errand* document should be news to the many USPTO decisionmakers that I have been trying to talk sense with during the past several years. This document merely collects all in one place the many points that I have been trying (with apparently limited success) to make to the USPTO for the past several years.

Details: In this document I have a goal of helping the reader understand a bit about how word processors work inside. I have a goal of helping the reader understand what it means for a word processor to store a data file in a particular format, and what assumptions if any a third party (such as a government agency) may reasonably make about that format. I have a goal of helping the USPTO to be smart, or at least to avoid being stupid, about how to receive patent applications in character-based format in a way that could attract cooperation and even buy-in from its customers (the applicants and practitioners who file patent applications).

State of play in 2002. We will begin the discussion by looking at Figure 1, and I will describe the “state of play” as it stood in the year 2002.



The way that it worked in 2002 is that there were two word processors with large market share, namely WordPerfect and Microsoft Word. There was also a distant-third-place word processor called OpenOffice which was nonprofit and open-source. Let's look at the first row to see the workflow and functional parts of a word processor.

The starting point with any word processor is that which is in the mind of the user. The user wants to write something. The user communicates this to the word processor (here, WordPerfect) through the user interface (UI) of the WordPerfect software. The software stores the work in a file on the hard disk of the computer, and for our discussion I will call this the "intermediate store". In 2002, the default filename extension used by WordPerfect for its "intermediate store" was "WPD". (This stood for "WordPerfect Document".) When the time came to print the document on a printer, or to view it on a computer screen, this task was carried out by means of what is called a "rendering engine". The rendering engine has the task of receiving as its input the intermediate store file and "rendering" the document in a human-readable form, for the purpose of putting ink on the page (if the target is a printer) or putting pixels on a screen (if the target is a screen). It is of great importance for the reader to understand that in 2002, with WordPerfect, the way this worked is that the rendering engine was a proprietary piece of software. The source code for this rendering engine was maintained in secrecy by the WordPerfect company, and the only code released to the user was executable code.

In this Figure 1, there are in fact *four boxes* that represent software that was released to the user only as executable code – the WordPerfect UI (editor), the WordPerfect rendering engine, the Microsoft Word UI (editor), and the Microsoft Word rendering engine. As for each of these four boxes, the corresponding source code was maintained in secrecy by its maker. The alert reader will of course know exactly where I am going with this -- the disruptor was OpenOffice, which provided its UI (editor) and rendering engine in both open-source code and in executable code. (I was an early adopter of OpenOffice.)

It is now of similarly great importance for the reader to join me in paying close attention to the standardized or non-standardized status of the three intermediate storage formats – WPD format, DOC format, and ODF (OpenDocument text) format. The state of play in 2002 was that the WPD format and DOC format were each a proprietary format. (The filename extension "DOC" is simply the first three letters of the word "document".) Of course enormous amounts of effort were expended by the maker of each word processor to try to reverse-engineer the proprietary formats of the other word processors. It was, after all, a matter of consumer survival for any word processor that was in second or third place to somehow accomplish interoperability with the dominant word processor. It was thus a matter of consumer survival for, say, WordPerfect to try as best it could to somehow export its intermediate-format files into DOC format. It was likewise a matter of consumer survival for WordPerfect to try as best it could to somehow import DOC-formatted files into the WordPerfect intermediate format. This was a never-ending game of cat and mouse. From time to time after WordPerfect had made progress in the reverse engineering, Microsoft would release another version of Microsoft Word in which a few more formatting codes were used to encode a few more kinds of rendered text according to some proprietary internal Microsoft standard. It is as if Microsoft had had a goal of defeating or at least impeding the reverse-engineering efforts at WordPerfect.

WordPerfect would then struggle to reverse-engineer the most recent DOC format so that it could hopefully import such files into WordPerfect while preserving such formatting within the WordPerfect environment, and so that it could hopefully successfully export such formatting from WordPerfect into the most recent version of the DOC format.

The breath of fresh air was OpenOffice and its OpenDocument format. (The "odt" filename extension stands for "Open Document text" format.) The OpenDocument format was explicitly established from the outset as a format based upon open published standards, administered by a neutral standards-setting body. The makers of OpenOffice also faced the same survival problem that WordPerfect faced, namely that there was no choice but to try as best they could to reverse-engineer the ever-shifting DOC format so that they could import DOC files and export DOC files. They also needed to be able to reverse-engineer the WPD format for similar reasons. (There came a time when this last need became less pressing for OpenOffice because eventually WordPerfect was driven off the market and ceased to be a meaningful competitor to Microsoft.)

The prospect of imports and exports of intermediate file formats is denoted in Figure 1 by the dashed lines labeled "Imports and exports". I use the dashed lines to signal to the reader that the imports and exports are very imperfect, with significant formatting errors and losses incurred in most of the imports and exports.

What have we learned (or reviewed) thus far? In about 2002, there were several proprietary intermediate storage formats, of which the two leading examples were WPD and DOC. Makers of word processor software had reached varying levels of success at reverse-engineering the various competing proprietary intermediate storage formats. There was an upstart open-source effort (OpenOffice) that threatened to disrupt the dominant market position of Microsoft. This

effort, had it succeeded, would have set into place an open-standards-based format for the intermediate storage of word processor documents, namely ODF.

What, in 2002, were Microsoft's goals in response to the then-recent entrance of OpenOffice and the OpenDocument format upon the word processor scene? The last thing Microsoft would want to see is all of the other word processor makers "ganging up" on Microsoft and migrating to a single commonly employed intermediate storage format, which format would be administered by some neutral standards-setting body. From the point of view of Microsoft, if such a migration were to happen, this would present the danger of competing word processors not only surviving against Microsoft but even perhaps thriving. One can see that Microsoft would benefit greatly if it could somehow derail the OpenDocument Text movement.

It was at this time that Microsoft devised a competing and supposedly "open" standard which it called DOCX, which was supposedly going to be administered by a different supposedly neutral standards-setting body.

Long-time Microsoft watchers predicted, correctly, the course of development. Microsoft managed to convince many players who had previously given their time and energy to the OpenDocument standard to stop supporting that effort, and instead to give their time and energy to the supposedly better DOCX "standard". (I put the term "standard" in quotation marks for reasons that will presently become clear.) Previous efforts to develop the OpenDocument standard to provide industry-wide open-source support for math formulas and chemical formulas lost momentum. Microsoft then engineered a "fork" in the development of the DOCX "standard" defining two branches. One branch (called "strict") would be the "open standards" branch that would be henceforth administered by the neutral standards-setting body, and the other branch (called "transitional") would be the branch actually supported by Microsoft Word going forward. The open-standards branch would then eventually go nowhere and would get resource-starved. The branch actually supported by Microsoft Word in an ongoing way would not be controlled by any neutral body but would be controlled by Microsoft, with changes to the format being made at whatever times Microsoft saw fit, and with limited publicly released documentation to whatever extent Microsoft saw fit to establish.

State of play in 2010. On a practical level the result was the state of play set forth in Figure 2, a state of play that largely persists to this day. WordPerfect is no longer a meaningful player. OpenOffice has been succeeded by an open-source fork called Libre Office. (I am a happy user of Libre Office.) The third major word processor in recent years has been Google Docs.

Figure 2.
State of play in 2010

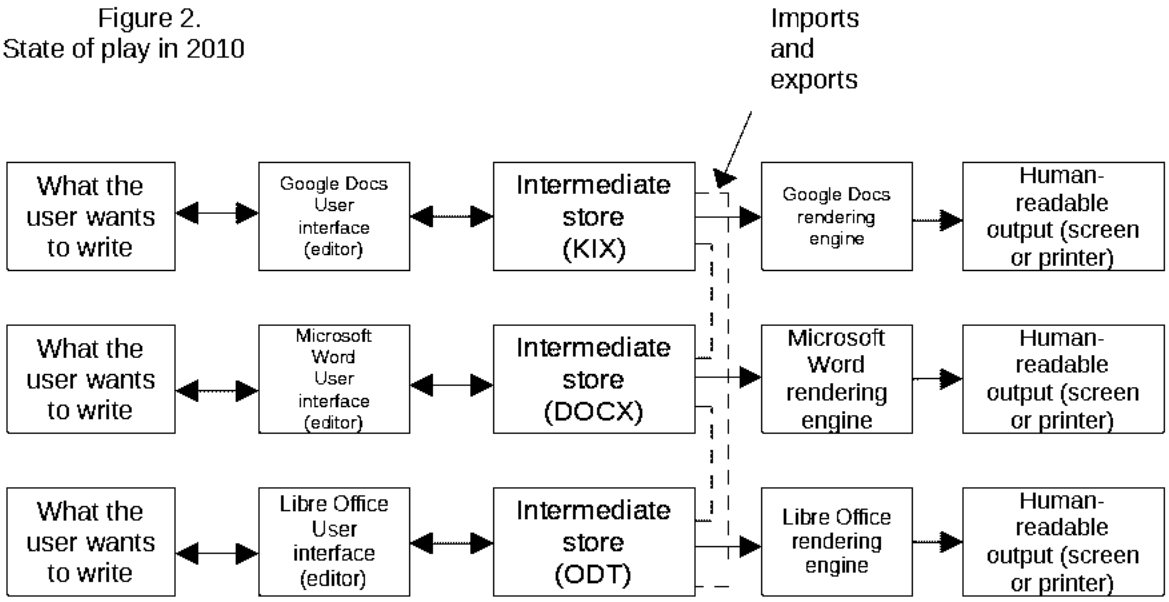


Figure 2 shows results that greatly favored Microsoft. A competitor (WordPerfect) is out of the market. The danger that two or more competing word processors might have migrated to make use of a single open-source intermediate format, administered by a neutral standard-setting body, has been averted.

The OpenDocument format (which continues to be denoted by the "odt" filename extension) continues to be completely "open". Part of the openness of the ODF ("odt") format flows automatically from the open-source nature of the present-day *UI* (editor) of Libre Office, together with the open-source nature of the UI (editor) of its ancestor, OpenOffice. Another part of the openness of the ODF format flows automatically from the open-source nature of the *rendering engine* of Libre Office, together with the open-source nature of the rendering engine of its ancestor, OpenOffice. All of the collaborators who participate in any way in versions or forks of OpenOffice or Libre Office publish information about any and all extensions and feature-adds that they incorporate into the ODF format.

A first consequence of this complete openness of the ODF format is that any other word processor maker (for example Microsoft or Google Docs) that chooses to try to ***export*** its own internal intermediate storage format (here, DOCX or KIX) into ODF is always able to do so with complete success and with no errors or artifacts or formatting failures. The only exception to such success would be the special case where some highly specialized or exotic type of formatting simply does not exist yet in ODF, in which case it is understandable that there would not be a way to make it happen in the export of a document into ODF format.

A second consequence of this complete openness of the ODF format is that any other word processor maker (for example Microsoft or Google Docs) that chooses to try to ***import*** an ODF file into its own internal intermediate storage format (here, DOCX or KIX) is always able to do

so with complete success and with no errors or artifacts or formatting failures. What never happens is that the programmer, parsing some ODF file, runs into some coding or string of characters that "makes no sense" or that the programmer "cannot figure out". There is always a clear and unambiguous way to make sense of every bit of content in an ODF file, drawing upon readily available publicly available information.

This last point bears emphasis. If there were ever some initially baffling data element in some ODF file where it appeared, at least superficially, that somehow the published standards documents somehow did not quite make completely clear how to interpret the data element, there is always a completely clear fallback position. The programmer can, if necessary, simply inspect the (publicly available!) source code of the word processor's rendering engine, and follow step by step exactly how the rendering engine renders the initially baffling data element into human-readable format. At this point, it will be clear simply by looking at how the data element got rendered into ink on the page, or by looking at how the data element got rendered into pixels on the screen, how the data element may be understood.

Another way to say this is that there is never a need for anyone to "reverse engineer" anything about the ODF format or the ODF standard. It is all open and open-source and it takes place according to published standards.

To reinforce the point of the previous paragraph in different words, what we have is that in Figure 1, the OpenOffice rendering engine is open-source, and in Figure 2, the Libre Office rendering engine is open-source, and this fact, together with the ODF community's shared values of documenting the standard, leads to a situation where there is never any lasting puzzlement about any detail of formatting. Any programmer who is connected with some other word processor, for example Google Docs or Microsoft Word, can always achieve an import or export to or from ODF with absolute and complete assurance of success and accuracy.

It is important to keep in mind that anyone can say that anything is supposedly "standards-based". Saying it does not make it so. As it turns out there are objectively measurable things that make it easy to determine whether someone is telling the truth or not when they say something is "standards-based", as I will discuss.

The presence of standards. Yes, this is part of how you know whether somebody is lying or telling the truth when they say something like a device or a data format is standards-based.

A first step for anyone to show that they are telling the truth when they claim that a device or a data format is standards-based is to point to some repository where the standards may be seen. If they cannot point to such a repository, or cannot point to the actual standards within the repository, this reveals the claim to be a lie.

Let's suppose the person points to some purported repository and the supposed standards within the repository. The second step for anyone to show that they are telling the truth when they claim that a device or a data format is standards-based is to point to various numbered and dated

versions of the standard that may be seen within the repository. If they cannot point to such numbered and dated versions of the standard, this reveals the claim to be a lie.

Let's suppose the person points to some purported repository and the supposed standards within the repository, and also points to various numbered and dated versions of the standard that may be seen within the repository. The third step for anyone to show that they are telling the truth when they claim that a device or a data format is standards-based is to point to evidence of recent standards-maintaining activity that has taken place. If the date of the most recent so-called "standard" is years or decades in the past, and if it is well known to users of the device or the data format that many changes have somehow happened in recent years, then this is a telltale that the device or data format is not in fact standards-based. This reveals the claim to be a lie.

There is a fourth thing that someone must do if they hope to show that they are telling the truth when they claim that a device or a data format is standards-based. They must point to some actual existing standard-setting body that maintains the repository and somehow facilitates ongoing standards-setting activity. Ideally the standard-setting body is an industry neutral, at arm's length from any for-profit participant. Failure to point to such an existing standards-setting body reveals the claim to be a lie.

It is easy enough to see that the OpenDocument format ("ODF") is standards-based. A moment or two of mouse-clicking in Wikipedia or Google quickly reveals that this standard is organized by the *Organization for the Advancement of Structured Information Standards* (OASIS) consortium. There is actually a most recent version of the standard, and it is version 1.3 which was adopted in January of 2020. There is a place (a repository) where you can click and view and download the standard.

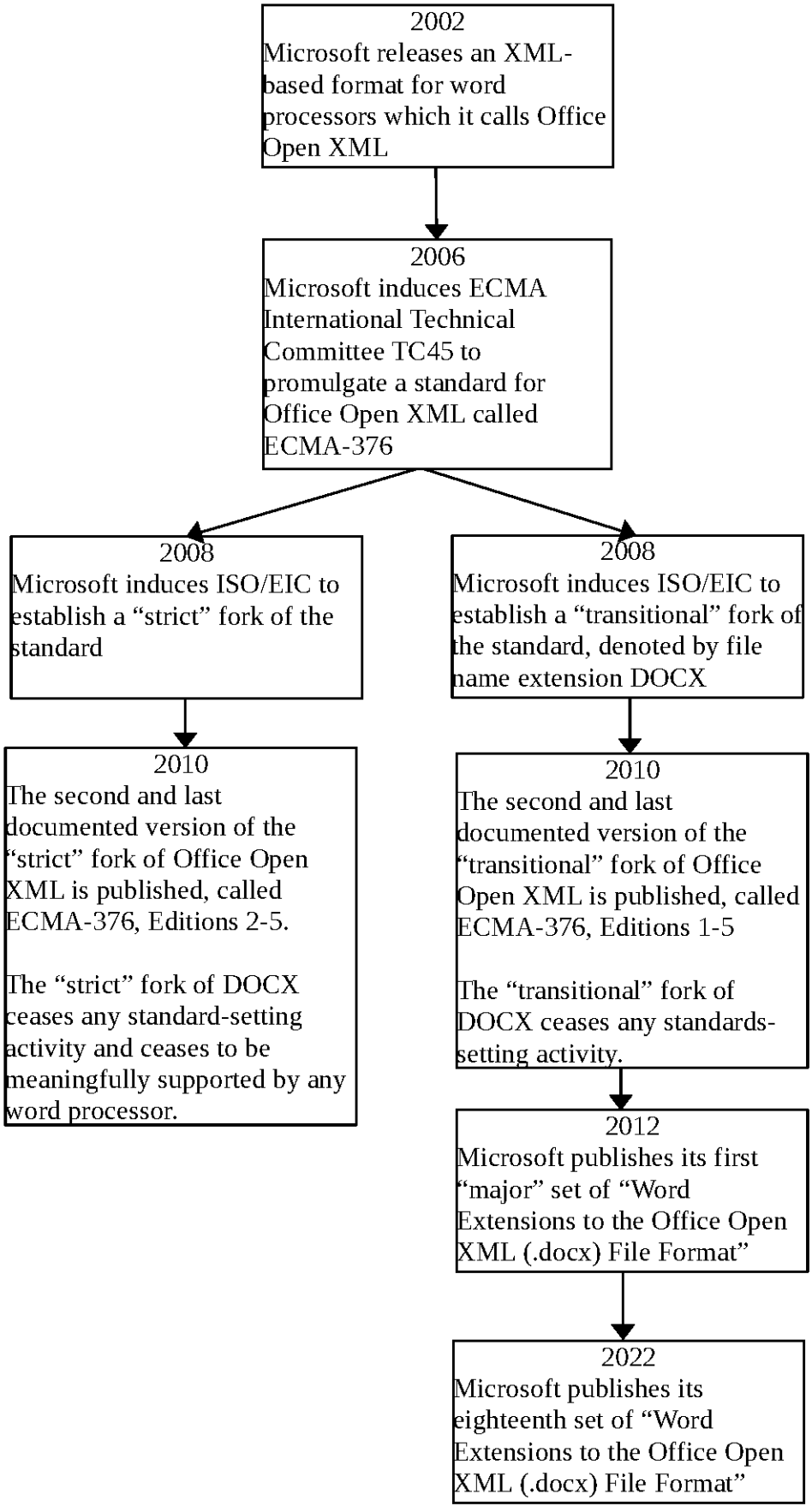
The USPTO surely knows perfectly well that the DOCX format as employed by Microsoft Word is not at all standards-based. Conspicuous by its absence in any USPTO documents or communications is any identification of a standard-setting entity for what the USPTO calls "the DOCX format". Conspicuous by its absence in any USPTO documents or communications is any identification of a repository for numbered and dated standards for what the USPTO calls "the DOCX format". The USPTO has not pointed, and cannot point, to any recent "version" of a supposed "standard" for what it calls "the DOCX format".

The history of the DOCX "standard". It is true that Microsoft's initial efforts to derail ODF and to set up a supposedly standards-based DOCX format did lead to standard-setting activity. One body called ECMA participated for a while, as did ISO. By 2008, there was a "fork" in the standard, a fork engineered by Microsoft, with one branch called "Strict" and the other called "Transitional". Only the "Strict" fork continued to have anything that remotely resembled an open standard-setting environment. The way it eventually worked out is that Microsoft Word, from about 2009 to the present, has supported only the "Transitional" fork which is not standards-based in any way. The "Strict" fork eventually became inactive. The most recent purported "release" of a new version of the "Strict" fork of DOCX was a "fourth edition"

released in 2016. No commercial software follows this “Strict” fork. No open-source software follows this “Strict” fork.

Figure 3 shows the history of the DOCX “standard”. It is recalled that in about 2002, some members of the word processing community established the OpenDocument standard for word processor files. As may be seen in Figure 3, Microsoft’s response was to create a competing “standard” in 2002 called Office Open XML, with a file name extension of DOCX. Microsoft induced two standards-setting bodies to lend authenticity to this “standard”, with ECMA promulgating a standard (long since fallen out of use) called ECMA-376 in 2006 and ISO promulgating a standard (also long since fallen out of use) ISO/IEC standard 29500:2008 in 2008. This standard embodies two variants or “forks”, a “strict” variant that had the support of the non-Microsoft participants and a “transitional” variant that was only supported by Microsoft.

Fig. 3. History of the DOCX “standard”



The variant of DOCX used in Microsoft Word (which has its origins in the "Transitional" fork of DOCX) last had a purported "standard" published in 2009 and 2010. Microsoft has not even pretended to conduct standard-setting activity for its fork of DOCX since 2010. There are no numbered or dated standards for its fork of DOCX (not since 2010). There is no standard-setting or standard-maintaining entity for standards for its fork of DOCX, at least, not a neutral party or a party at arm's length from Microsoft.

What Microsoft does publish is opaque documents called "Word Extensions to the Office Open XML (.docx) file format". There have now been more than eighteen of these "extensions" to the DOCX standard as it existed in 2008 and 2010.

To be clear about this, the so-called DOCX standard ceased to have any standard-setting activity by 2010. What has happened since 2010 is "extensions" unilaterally imposed by Microsoft, with no participation by any other entity. The flavor of DOCX now in use in Microsoft Word is and continues to be a "moving target" for any word processor provided by anyone other than Microsoft.

Let's return now to false and disingenuous statements on the USPTO web site. For example the USPTO says this on its web site:

> The DOCX format is an international standard defined under ECMA-376 and ISO/IEC 29500 and approved by the Library of Congress.

If you look at Figure 3, you can see that the last time any standard-setting took place under ECMA-376 was twelve years ago. No present-day word processor follows that ECMA-376 standard from 12 years ago.

If you fact-check the USPTO about the ISO/IEC 29500, you find that the last time any standard-setting activity took place under that standard was in 2016 (more than six years ago). No present-day word processor follows that ISO/IEC 29500 standard from more than six years ago. Instead, Microsoft has "extended" its flavor of DOCX some eighteen times since then.

The USPTO is using the existence of those old standards from six years ago and older, which no one follows nowadays, to try to legitimize its present-day efforts to force patent applicants to use its present-day proprietary PDF rendering engine and proprietary DOCX validating and processing software. This is deceptive on the USPTO's part, and is intellectually dishonest.

The USPTO's citation to the Library of Congress is amusing for what it fails to say. The Library of Congress also approved the ODF format! See https://www.loc.gov/preservation/digital/formats/fdd/fdd000247.shtml .

To be thorough about all of this we must now remind ourselves which boxes in Figure 2 are proprietary. Although it is not particularly important to the things that we are discussing in this

article, it happens that Google does publish details of its internal storage format (KIX). Google also goes to extraordinary lengths to use some of its best and brightest people to achieve the highest possible levels of interoperability between Google Docs and all of the non-Microsoft word processors that remain in business. The ODF exports from Google Docs, for example, are nearly always nearly a complete success when a Libre Office user opens them in Libre Office, with vanishingly few formatting mistakes or losses. Similarly, when a Google Docs user imports an ODF file that had previously been created by a Libre Office user, the result in Google Docs is nearly always nearly a complete success, with vanishingly few formatting mistakes or losses.

These interoperability successes as between Google Docs and Libre Office (and other word processors with even smaller present-day market shares) are something to feel good about in a world where a dominant player (here, Microsoft) might consider interoperability to be a bug rather than a feature. This successful interoperability between non-Microsoft word processors is due in large part to the simple fact that the OpenDocument text format is a published and open standard.

As we work our way through Figure 2 in our review of which boxes are proprietary and which are not, our gaze focuses on two boxes which now require clear-eyed study because this bears on USPTO's recent choices about how to receive patent applications.

A first important box is the Microsoft Word rendering engine which renders its DOCX internal format into human-readable ink on the page or renders its DOCX internal format into human-readable pixels on a screen. This rendering engine is provided to users only as executable code. The source code for this rendering engine is closely guarded source code within Microsoft.

A second important box is the "intermediate store (DOCX)" box. What this box represents is the fork of DOCX that Microsoft ended up choosing to "extend" (by now eighteen times) in its present-day Microsoft Word software. This is a fork of DOCX that, importantly, is not administered by any neutral standard-setting body. The internal structure of a DOCX file, if it got created by Microsoft Word, is a moving target. It might have gotten created by a version of Microsoft Word from a year ago, running on a version of Microsoft Windows. If so, it might have a first internal structure. It might, on the other hand, gotten created by a version of Microsoft Word from a month ago, running on a version of an Apple Mac operating system. If so, it might have a second internal structure. Microsoft provides some level of documentation of the various DOCX "extensions" that it employs, but Microsoft does not document any of the DOCX formats thoroughly. Importantly, Microsoft does not promise to give any particular warning or advance notice when it chooses to make yet another "extension" to its flavor of DOCX format.

Related to this is the reality that Microsoft can, and often does, make version changes to its Microsoft Word rendering engines from time to time. When Microsoft makes a version change to one of its rendering engines (for example in an Apple Mac or in a Windows environment), an external observer is not in a position to know exactly what may have changed in the function of the rendering engine.

Even now in 2022, any maker of a word processor that is not Microsoft Word, that sets a goal of ***exporting*** an intermediate-storage file into a DOCX format with the goal that it will be opened by a user of Microsoft Word, faces challenges in implementing such a goal. Even now in 2022 there is sometimes a need to carry out reverse engineering on the DOCX-formatted files that are generated by Microsoft Word in an effort to work out how to construct a DOCX file that will work as desired when it gets opened by Microsoft Word. Importantly, an export that worked in January of a given year might not work in February of that year because of some change that Microsoft made between January and February in its proprietary UI (editor) or in its proprietary rendering engine. Because Microsoft does not give advance notice of such changes, it frequently happens that the maker of the non-Microsoft word processor finds out about the change "the hard way" simply by finding out that in February the export failed even though it had worked in January.

Even now in 2022, any maker of a word processor that is not Microsoft Word, that sets a goal of ***importing*** an intermediate-storage file into its own internal storage format, drawing upon a DOCX file that had been created by a user of Microsoft Word, faces challenges in implementing such a goal. Even now in 2022 there is sometimes a need to carry out reverse engineering on the DOCX-formatted files that are generated by Microsoft Word in an effort to work out how to parse a DOCX file so as to make sense of it during the import process. Importantly, an import that worked in January of a given year might not work in February of that year because of some change that Microsoft made between January and February in its proprietary UI (editor) or in its proprietary rendering engine. Because Microsoft does not give advance notice of such changes, it frequently happens that the maker of the non-Microsoft word processor finds out about the change "the hard way" simply by finding out that in February the import failed even though it had worked in January.

It is true that every maker of a non-Microsoft word processor has no choice, as a matter of survival, but to try as best it can to ***export*** its own files into Microsoft's version of the DOCX format. Google Docs does so as best it can. Libre Office does so as best it can. Such exports are, however, rarely a complete success except when the word processing file is very simple. If there is anything at all complicated about the word processing file being exported, it nearly always turns out that some formatting is corrupted or lost. It nearly always turns out that page breaks are in different places and math and chemical formulas look at least slightly different. Strikingly often, some seemingly small and simple bit of formatting will look astonishingly different when viewed in Microsoft Word as compared to the seemingly identical bit of formatting in the original non-Microsoft word processor.

It is likewise true that every maker of a non-Microsoft word processor has no choice, as a matter of survival, but to try as best it can to ***import*** files that were created in Microsoft Word (files that are formatted in Microsoft's version of the DOCX format). Google Docs does so as best it can. Libre Office does so as best it can. Such imports are, however, rarely a complete success except when the word processing file is very simple. If there is anything at all complicated about the word processing file being imported, it nearly always turns out that some formatting is corrupted

or lost. It nearly always turns out that page breaks are in different places and math and chemical formulas look at least slightly different. Strikingly often, some seemingly small and simple bit of formatting will look astonishingly different when viewed in the new non-Microsoft word processor as compared to the seemingly identical bit of formatting in the original Microsoft Word word processor.

What USPTO says falsely about DOCX. With the benefit of the above discussion, we can now assess a statement on the USPTO web site. The USPTO says:

> **What is DOCX?**
>
> DOCX is a word processing file format based on open standards, including Extensible Markup Language (XML). DOCX is supported by many popular word processing applications, such as Microsoft Word 2007 or higher, Google Docs, Office Online, LibreOffice and Pages for Mac. As an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents.

Pretty much everything in this quoted paragraph is pants-on-fire false. The most charitable way to characterize this quoted paragraph would be to say that everything in it is disingenuous in the extreme. Let's take the false statements in this paragraph one by one.

> DOCX is a word processing file format based on open standards, including Extensible Markup Language (XML).

One of the falsehoods (or charitably, misleading aspects) in this sentence is the use of the singular "a". There is no single "DOCX word processing file format". There is the fork that ended up going nowhere (called "Strict"), which arguably was based on open standards. There is the fork that ended up actually remaining in commercial use (called "Transitional" as modified by Microsoft's by now eighteen "extensions" to the DOCX format), and it is not now an open standard.

It might be possible to edit this sentence until, eventually, it became a true statement. One way to do it would be to say something like:

> One of the forks of DOCX is the fork that is a word processing file format used by Microsoft Word. Although the Microsoft Word fork of DOCX is not now an open standard, it did historically draw upon open standards, including Extensible Markup Language (XML). Microsoft has "extended" its version of DOCX eighteen times by now and bears little resemblance to the most recent actual DOCX standard, which was promulgated twelve years ago.

Now we can work on the second sentence:

> DOCX is supported by many popular word processing applications, such as Microsoft Word 2007 or higher, Google Docs, Office Online, LibreOffice and Pages for Mac.

Probably the chief evil in this sentence is the attempt to force-fit the word “supported” into a dual role, with that single word on the one hand characterizing the connection between DOCX and Microsoft Word, and on the other hand characterizing the connection between DOCX and all of the other word processors.

Google Docs does not in any meaningful sense “support” the Microsoft Word DOCX format, except in the limited sense that as a matter of survival it has no choice but to try as best it can to *import* Microsoft Word DOCX files and *export* its files into Microsoft Word DOCX format.

Libre Office does not in any meaningful sense “support” the Microsoft Word DOCX format, except in the limited sense that as a matter of survival it has no choice but to try as best it can to *import* Microsoft Word DOCX files and *export* its files into Microsoft Word DOCX format.

On the other hand, of course Microsoft Word “supports” its own DOCX format! The proprietary UI (editor) of Microsoft Word exists specifically so as to create files in its own DOCX format. The proprietary Microsoft Word rendering engine exists specifically so as to render its own DOCX-formatted files into human-readable pixels on a screen, and to render its own DOCX-formatted files into ink on the page.

One way to edit that sentence into a non-false sentence would be like this:

> DOCX is the internal storage format used by Microsoft Word. Many popular non-Microsoft word processing applications, including Google Docs, Office Online, LibreOffice and Pages for Mac, try to *export* their own internal storage formats into DOCX files which can then be opened by users of Microsoft Word with varying degrees of success. These non-Microsoft word processing applications likewise try as best they can to *import* files created by Microsoft Word (in its DOCX format) into their own respective internal storage formats. For very simple files containing only the very simplest formatting, the imports and exports are often generally fairly successful. When the file being imported or exported contains formatting of any complexity, it is rare that the import or export is completely successful. The DOCX internal storage format used by Microsoft Word changes from time to time due to ongoing “extensions”, and thus as a consequence an import or export to or from a non-Microsoft word processor that worked on some particular date might not work a month later.

We can now return to the third sentence in the USPTO paragraph:

> As an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents.

Everything about this sentence is pants-on-fire false. First, to the extent that one makes use of DOCX as a practical term, it is false to say that it is "an open standard format". One can talk about "the DOCX-formatted files that get created by Microsoft Word". One can talk about "the DOCX-formatted files that you get when you export from Google Docs". One can talk about "the DOCX-formatted files that you get when you export from Libre Office". Each of these three sets of files is formatted in its own way. There is no single "open standard" that defines how the three sets of files are formatted. There was an open standard for DOCX in 2010. Twelve years, however, have passed since the last time there was an "open standard" for DOCX.

Part of the disingenuousness of this sentence flows from the fact that the only way to get meaning from an intermediate storage file of a word processor is to render it into human-readable form. Only when the file has been rendered into human-readable form (for example as a patent application visible by a human being as pixels on a screen, or for example as a patent appication visible by a human being as ink on a page) is anyone able to know what is communicated in, say, claim 1 of the patent application.

How exactly does the rendering take place? For example, suppose the patent application has been stored on the hard drive of a computer with a filename extension of DOCX. How may we render it into human-readable form, for example as ink on the page on a printer?

Returning to Figure 2, we can easily work out three possible workflow paths for rendering the DOCX file into human-readable form on a printer. We need to be methodical about enumerating the three workflow paths:

- Use Google Docs.
  - For this path, we start by going to Google Docs and we carry out an "import" of the DOCX file into the internal storage format of Google Docs. (As mentioned above, it is called "KIX" but for our purposes of this discussion the name of the internal format is not important.)
  - Having imported the DOCX file into the internal storage format of Google Docs, we then run the Google Docs rendering engine to render the KIX document into human-readable form on the printer.
  - We end up with ink on the page.
- Use Microsoft Word.
  - For this path, we run the Microsoft Word rendering engine to render the DOCX file into human-readable form on the printer.
  - We end up with ink on the page.
- Use Libre Office.
  - For this path, we start by going to Libre Office and we carry out an "import" of the DOCX file into the internal storage format of Libre Office, namely ODF format.
  - We then run the Libre Office rendering engine to render the ODF document into human-readable form on the printer.
  - We end up with ink on the page.

We can reflect upon these three paths to consider what there is about the three paths that is proprietary in nature, and where (if at all) we might encounter sources of error or formatting loss.

Let's start with the third path (Libre Office). Most of this path is rather predictable in the sense that the software itself is open-source. The "import" gets carried out by open-source software. The rendering engine is open-source software. The chief source of error and formatting loss is the unpredictable behavior of Microsoft. By this we mean that even if the programmers of Libre Office had somehow successfully done all of the reverse engineering needed to get all of this to work successfully, say, a month ago, what might have happened during the past month is that Microsoft may have made yet another "extension" to its flavor of DOCX. If Microsoft happens to have done so, then it is unlikely to have documented the format change in any methodical or thorough way. Even if Microsoft did document the format change to some extent, Microsoft is unlikely to have communicated the documentation change to others in any thorough way.

Now let's turn to the first path (Google Docs). Just as with Libre Office, an important source of error and formatting loss is the unpredictable behavior of Microsoft. Just as with Libre Office, even if the programmers of Google Docs had somehow successfully done all of the reverse engineering needed to get all of this to work successfully, say, a month ago, what might have happened during the past month is that Microsoft may have changed something about how it formats its DOCX files. This would trip up the Google Docs file import just as it would trip up the Libre Office file import.

A second source of possible unpredictability in this path, from the end user point of view, is that the import software that imports the DOCX file into the internal Google Docs format is proprietary. Google might carry out a version change to its import software on some particular date and the end user might not know that this version change had happened. It cannot be ruled out that some change in the import software might change something about how the DOCX file gets converted into the Google Docs format, which would necessarily change how the file gets rendered into human readable form on the printer.

A third source of possible unpredictability in this path, from the end user point of view, is that the Google Docs rendering engine is proprietary. Google might carry out a version change to its Google Docs rendering engine on some particular date and the end user might not know that this version change had happened. It cannot be ruled out that some change in the Google Docs rendering engine might change something about how a word processor file gets rendered into human readable form on the printer.

Now we can turn to the remaining flow path (the second path, using Microsoft Word). The idea here, simply put, is that we are using a document that probably got created using Microsoft Word, and we are, oddly enough, using Microsoft Word to render the document into human-readable form, for example on a printer. The alert reader will be able to guess where I am going with this. If there were ever a flow path that is *unlikely* to blow up in the user's face, it is the flow path in which you use the same word processor to print the file that had previously been employed to create the file.

The discussion of the preceding four paragraphs focuses on the part where someone has a DOCX file and they are going to try to send it to a printer. What no one at the USPTO has paid any attention to is that if you try these three paths to send any single DOCX file to a printer, you will absolutely never get identical results from the three paths. Hyphenations of words will happen in different places. Superscripts and subscripts will lead to non-identical variations in line spacings. Page breaks will happen in non-identical places. These kinds of differences might not seem to be material, but they vividly remind the user that if these kinds of things do not come out identically, then other more subtle things like mathematical equations or chemical formulas or tabular presentations of data, or special characters or Greek letters, would very likely also not come out identically. Users have seen differences in DOCX rendering across various word processors that lead to a need for character-by-character proofreading that takes hours.

Now let's return to the quoted USPTO sentence:

> As an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents.

The only word processor in which a user can "author" a DOCX file is Microsoft Word. Let's focus on the "authoring" claim. Let's suppose you want to "author" a DOCX file in Libre Office. What the USPTO is in seeming denial about here is that it is impossible to "author" a document in DOCX format with any word processor that is not Microsoft Word. This by itself makes that USPTO sentence false.

Suppose you set a goal of "authoring" a DOCX file in any word processor that is not Microsoft Word. The way you do this is by "authoring" the document in whatever internal storage format the word processor uses (for example KIX or ODF). And then you run the "export" function so that the word processor will do its best at generating a word processor file that ends in the letters "docx" and will hopefully be more or less successful when opened by Microsoft Word.

This bears emphasis and repetition:

> *The only word processor in which a user can "author" a DOCX file is Microsoft Word. If you are using a word processor that is not Microsoft Word, you cannot "author" a DOCX file. The closest you can get to this is "authoring" the document in whatever internal storage format the word processor actually uses and then "exporting" the file as best you can into a format that is intended to work when opened in Microsoft Word.*

Anyone who works at the USPTO and who has some connection with this DOCX disaster should read the preceding paragraph over and over again until they understand it clearly.

The risk factors and sources of unpredictability in attempting to "author" a DOCX file when using a non-Microsoft word processor are rather similar to those that we discussed for the printing of the DOCX file in Libre Office, *mutatis mutandis*. To "author" a DOCX file in Libre

Office, we start by using the UI (editor) to capture our thoughts and to store them in ODF format. This uses software for which the open-source code may be viewed. Nothing about it is unpredictable. We then carry out the *export* into the Microsoft Word (DOCX) format. This, too, uses software for which the open-source code may be viewed. Nothing about it is unpredictable, other than in the sense that Microsoft may have fiddled with ("extended") its DOCX format in the weeks or months that passed since the last time that the Libre Office programmers did their reverse engineering to see what the latest changes had been in the Microsoft Word DOCX formatting. Importantly, the resulting DOCX file is meaningless when taken by itself in isolation. It means something only when we realise that to do something with it, somebody is going to need to pass the DOCX file through the Microsoft Word proprietary rendering engine. Here, too, we are at the mercy of unpredictable Microsoft behavior. Microsoft may well have fiddled with its rendering engine since the last time that people outside of Microsoft had an opportunity to explore the behavior of the rendering engine. All of these things lead to a less than completely predictable authoring process for the Libre Office user that decides to take a chance and try to author a DOCX file.

It is simply pants-on-fire false for USPTO to characterize this as "safe and stable". The Libre Office user who sets a goal of "authoring" a DOCX file for use as a patent application at the USPTO has neither safety nor stability, given that the fork of DOCX being used by Microsoft Word is not standardized in any public way and could change at any time, and given that the USPTO's own proprietary rendering engine for rendering DOCX files into PDF could change at any time. The same is true for the Google Docs user who sets such a goal; it is neither safe nor stable.

So we return again to the USPTO sentence:

> As an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents.

The sentence is false, and as it relates to "authoring" for any user of a word processor that is not Microsoft Word, the sentence is a howler. How might we edit the sentence to make it into a true statement about authoring? Some options include:

> Microsoft Word offers a safe and stable basis for authoring and processing intellectual property documents in USPTO's DOCX e-filing initiative, except of course for the problem that the USPTO's own proprietary DOCX rendering engine changes from time to time without warning.
>
> DOCX fails to offer a safe or stable basis for authoring and processing intellectual property documents with respect to word processors other than Microsoft Word.

The non-standard status of Microsoft's version of DOCX is well known. See for example *Complex singularity vs. openness,* at https://joinup.ec.europa.eu/sites/default/files/document/2014-06/complex_singularity_vs_openes

s.pdf . This document describes Microsoft's manipulation of the standards-setting process and the present-day consequences.

> When it comes to office documents, public administrations can choose from two ISO/IEC standards. Only one of these, ODF (ISO/IEC 26300), is vendor-neutral, open and reliable across a span of years and software versions, and supported by a variety of software products.
>
> The later OOXML standard (ISO/IEC 29500), originally developed by a single proprietary software vendor [Microsoft], is implemented in three different versions ('ECMA', 'Transitional' and 'Strict') that are not compatible with each other. Although the 'ECMA' and 'Transitional' versions are outdated – 'Transitional' had only been accepted as a temporary solution to give the software vendor [Microsoft] time to implement 'Strict' in its products – they both continue to be used in practice. This is because older versions of the vendor's office suite (MS Office) cannot read or write OOXML Strict and are unlikely ever to gain such abilities.
>
> … to date there are no free and open source solutions that fully support OOXML.
>
> Experts have shown that public administrations should not rely on ISO 29500 when exchanging documents, as this is likely to create ambiguities when using office tools that do not fully support ISO 29500 ...
>
> Many of the features in ISO 29500 are tied to versions of the proprietary office suite [Microsoft Word], reflecting this software's history and development decisions.
>
> ... since the 'Strict' [OOXML] standard used by Microsoft is still neither fully documented nor open (it contains references to Microsoft websites, some of which no longer exist), data loss on conversion is a widespread and well-documented phenomenon.
>
> By establishing its Markup Compatibility and Extensibility (MCE) technology in ISO 29500 Microsoft has gained the right to make changes to the document format simply by adding their own extensions, almost without limits. That makes it hard for any other company or open source project to be fully compatible.

It is well known within the word processor standards community that Microsoft's actions thwarted any meaningful standards-setting. See *Norwegian standards body implodes over OOXML controversy*, October 3, 2008, Ars Technica (https://arstechnica.com/uncategorized/2008/10/norwegian-standards-body-implodes-over-ooxml-controversy/):

> Standards Norway, the organization that manages technical standards for the Scandinavian country, took a serious blow last week when key members resigned in

> protest over procedural irregularities in the approval process for Microsoft's Office Open XML (OOXML) format. The 23-person technical committee has lost 13 of its members.
>
> The standardization process for Microsoft's office format has been plagued with controversy. Critics have challenged the validity of its ISO approval and allege that procedural irregularities and outright misconduct marred the voting process in national standards bodies around the world. Norway has faced particularly close scrutiny because the country reversed its vote against approval despite strong opposition to the format by a majority of the members who participated in the technical committee.

Wikipedia lists multiple controversies that arose during Microsoft's manipulation of the international standards-setting process for DOCX (https://en.wikipedia.org/wiki/Standardization_of_Office_Open_XML#Complaints_about_the_national_bodies_process). There are reports of Microsoft allegedly improperly influencing the portions of the standards-setting process that took place in 2008 in several countries:

- Portugal
- Sweden
- Finland
- Switzerland
- Australia
- Germany
- Netherlands
- Poland

It is embarrassing to see the USPTO pointing to the very tainted OOXML standards-setting process for DOCX format in 2008 as somehow legitimizing the USPTO's present plan of forcing patent applicants to use a present-day Microsoft variant of the DOCX format for the filing of patent applications.

The lie in the August 3, 2020 Federal Register notice. In the August 3, 2020 FR notice, the USPTO said this in its response to Comment 59.

> *Comment 59:* Two commenters stated that there is no single DOCX standard to which Microsoft Word and the other word processors are all compliant.
>
> *Response:* DOCX is a word-processing file format that is part of Office Open XML (OOXML), an XML-based open standard approved by the Ecma International® consortium and subsequently by the ISO/IEC joint technical committee. For more information about the OOXML standard, please see:
>
> - ECMA–376 at http://www.ecmainternational.org/publications/standards/Ecma-376.htm
> - ISO/IEC 29500 at https://www.iso.org/committee/45374/x/catalogue/

- NIST votes for U.S. Approval of OOXML at https://www.nist.gov/news-events/news/2008/03/nist-votesus-approval-modified-office-openxml-standard

Fact-checking these claims reveals the following:

- The ECMA standard ECMA-376 is more than six years old, and no current word processor follows that standard. Microsoft has "extended" its flavor of DOCX some eighteen times since the last time any standards-setting activity took place for the ECMA-376 standard.
- The ISO/IEC 29500 standard is more than six years old, and no current word processor follows that standard. Microsoft has "extended" its flavor of DOCX some eighteen times since the last time any standards-setting activity took place for the ISO/IEC 29500 standard.
- The NIST vote happened fourteen years ago. The version of the standard that NIST voted for is fourteen years old. No current word processor follows the standard that NIST voted for fourteen years ago. The variant of the standard that NIST voted for was what eventually became the "strict" fork of the standard, which never got implemented in any word processor.

The truthful USPTO response to Comment 59 would have been "yes, you are right, there is no single DOCX standard to which Microsoft Word and the other word processors are all compliant." Saying this differently, everything in the USPTO's response to Comment 59 was a lie.

Everybody knows there is no single DOCX format. Everybody except the USPTO, I guess. (I think the USPTO knows this too but cannot admit it because this would expose previous USPTO statements about this to be lies.) Every user of Libre Office, for example, encounters daily reminders that there is no single DOCX format. You can take a DOCX file that was created using, say, Microsoft Word, and when printed on a printer using Microsoft Word, it is, say, sixteen pages long. That exact same DOCX file, opened in Libre Office and printed to the exact same printer, might be fifteen pages long or seventeen pages long.

The same is true for Google Docs and DOCX. It is rare that any particular DOCX file that has more than ten pages will yield the exact same page count, when printed using all three commonly used word processors.

From these simple and objectively confirmable results, it is clear to any thinking person that it must be a lie to say that there is some single "DOCX standard" or some single "DOCX format".

You can take a multipage document created in any one of these three word processors, and then open the document in either of the other two word processors, and what you will see is that line breaks happen in non-identical places and page breaks happen in non-identical places. Hyphenations at ends of lines of text will happen in non-identical places.

From such simple and objectively confirmable results, any thinking person would say that there is no single “DOCX standard” or some single “DOCX format”.

USPTO has failed to publish the source code for its PDF rendering engine, or to explain its provenance. Practitioners who have used USPTO’s DOCX e-filing system are familiar with the process. The practitioner uploads a word processor file whose file name ends with the letters “docx”. The USPTO does not trust that file to be the authoritative file, but instead runs that file through USPTO’s proprietary PDF rendering engine. The resulting PDF file is presented to the practitioner. It is up to the practitioner, in the remaining minutes between now and midnight, to attempt to detect the ways that the USPTO rendering engine may have damaged the word processor file. Maybe a Greek letter μ got changed to an m. Maybe a mathematical equation got corrupted. Maybe a chemistry formula got corrupted. To be granted a filing date, the practitioner is required to check a box agreeing to an adhesion contract providing that the USPTO-generated PDF file “controls”.

It is more than two years ago that I first asked high-up USPTO people to publish the source code for its PDF rendering engine. If USPTO had done so, then this would have gone a long way toward legitimizing the USPTO’s adhesion-contract approach.

Yet another question is the provenance of the code for USPTO’s PDF rendering engine. Some practitioners suspect that the code for USPTO’s PDF rendering engine came from Microsoft. It is further suspected that this transaction was not at arm’s length. I have asked high-up USPTO people over and over again where exactly they got their PDF rendering engine, and have never gotten an answer.

A related problem is that in the most recent version of USPTO’s e-filing workflow for DOCX patent applications, the DOCX file gets “processed” by a proprietary USPTO DOCX validation engine. The filer uploads a DOCX file, and the USPTO passes the DOCX file through its proprietary USPTO validation engine, and a modified version of the DOCX file results. The USPTO then lies in the Acknowledgment Receipt and falsely indicates that the modified DOCX file is what the filer uploaded initially. (There is no softer word than “lies” that accurately characterizes this part of the DOCX e-filing process.)

I have asked USPTO people to publish the source code for USPTO’s DOCX validation engine. USPTO has declined to do so. The Director says that this DOCX validation engine is now in “version 18”, as if this were a good thing. It is of course a profound source of continued uncertainty and anxiety for filers, given that there will inevitably be a version 19 and a version 20, and they will get placed into service without advance warning and without a public change log.

Does the USPTO really believe its own stated position that there is a present-day DOCX standard? Actions speak louder than words. The *words* from the USPTO are:

> As an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents.

Those are USPTO's *words*. But let's look at the *actions* of the USPTO. When the USPTO initially rolled out its program for e-filing of patent applications, the program that uses the initialism "DOCX" over and over again, did the USPTO's actual actions show that the USPTO believed this? The clear answer is "no". The USPTO did not, for example, set up the DOCX pilot program like this:

- Applicant e-files a DOCX file.
- USPTO trusts that the DOCX file will work for the USPTO's purposes.

Instead, the way that the USPTO initially set up the DOCX pilot program was:

- The filer e-files a DOCX file.
- The USPTO absolutely does not trust the DOCX file for any purposes.
- The USPTO, in real time, during the e-filing process, runs the DOCX file that the applicant e-filed through a proprietary, black-box USPTO rendering engine into a PDF file.
- The filer is required to try to figure out whether or not the rendering engine caused harm to the document when it generated the PDF.
- To get a filing date for the filing, the filer is required to click to agree to an adhesion contract saying that "the PDF file controls".

What we see in the actions of the USPTO was a tacit admission that DOCX *does not* offer a safe and stable basis for authoring patent applications, because the only thing the USPTO trusts is what comes out of the PDF rendering engine, not the DOCX file itself.

A polite way to put this would be to say that the USPTO was and is being extremely disingenuous about whether or not there is "a DOCX standard". When it is convenient to pretend that there is "a DOCX standard", for example in USPTO's webinars that try to convince filers to use the DOCX pilot program, then USPTO says it believes that there is "a DOCX standard". But in USPTO's own actions, which again speak louder than words, it is revealed that the USPTO does not actually believe what it says about this. The only thing that the USPTO trusts is the PDF that gets spit out from the USPTO's own proprietary validation and rendering engine.

A more direct way to put this is that from USPTO's own actions, it is clear that the USPTO knows perfectly well it is lying when it says things like:

> As an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents.

USPTO does not even pretend to follow any "DOCX standard". Let's suppose there were "a DOCX standard" in 2022 (which there is not, but let's pretend the USPTO is telling the truth when it claims that there is "a DOCX standard" in 2022). Does the USPTO allow the filer to rely upon that present-day DOCX standard? The answer is no, the USPTO actually requires the filer to comply with a poorly defined USPTO variant of "the DOCX standard". The poorly defined variant is what you get if you start with this (nonexistent) "DOCX standard" and graft onto it some additional requirements imposed by the USPTO. These include for example that t*he filer is only permitted to make use of a short list of 28 permitted fonts. (*Offensively, the document from the USPTO that enumerates the 28 permitted fonts also says the list of permitted fonts "may be subject to change without notice".) The poorly defined variant also includes a requirement that the filer craft the DOCX file to avoid any of a range of validation errors in a proprietary (not open-source) validation engine inside the USPTO's e-filing system. In the Director's Blog, she states with some pride that the validation engine "is now at a very advanced stage (version 18)." This is actually something to apologize for, rather than to brag about. It means that by now the actual required non-standard variant of DOCX is *eighteen times removed* from whatever the supposedly industry-standardized DOCX format was when this DOCX initiative began.

And we must assume that there will some day be a version 19 of this proprietary DOCX validation engine, and after that, a version 20. The variant of (supposedly industry-standard) DOCX that USPTO will accept from filers will change again, and it will change again after that.

A chief purpose of USPTO's DOCX validation engine. Any time a filer uploads a DOCX patent application, the USPTO system runs the file through its proprietary DOCX validation engine. The system then presents the "validated" DOCX file to the filer and it is then up to the filer to try to guess what harm has been visited upon the word processor document.

It looks to me as though a chief purpose of USPTO's DOCX validation engine is to test whether the DOCX patent application was created using Microsoft Word (which is of course what the USPTO wants but cannot say openly) or was instead exported from some non-Microsoft word processor (which is of course what the USPTO does not want but cannot say openly).

When a practitioner uses Microsoft Word, and keeps the patent application simple (no math equations, no tables, no chemistry formulas, no Greek letters), it is commplace for the "validated" DOCX file to announce that there are "no errors".

In contrast, when a practitioner uses a non-Microsoft word processor and exports the document into DOCX format, the usual result of "validation" by the USPTO validation engine is a string of error reports with as many errors as there are paragraphs in the document.

A chief purpose of the USPTO DOCX validation engine appears to be a not-so-subtle nudge that the practitioner should spend the money to purchase Microsoft Word.

Selecting a word processor intermediate-storage format for USPTO patent application filing. It is instructive at this point to return to Figure 2 and to remind ourselves of USPTO's stated goal in designing its word-processor based initiative for receiving US patent applications in a character-based format. Let's suppose we take the USPTO at its word on what its selection criteria were. What we see is that what USPTO said is that it was trying to figure out if there is an intermediate-storage word processor format that satisfies several requirements:

- the format is actually standards-based right now
- the standard is really an open standard
- every word processor is able to either save in that format or export into that format
- the saving or exporting is stable (likely to work the same way tomorrow that it works today)

What happened next is that the USPTO said that DOCX is the answer. The pesky problem with this is that this is false, four times over. DOCX actually fails all four conditions.

The fork of DOCX actually in use now (in Microsoft Word) is a divergence from the "Strict" standards-based fork of DOCX. It is also eighteen times (by now) removed even from the "Transitional" fork of the standards-based DOCX standard. The Microsoft Word fork of DOCX ceased to be standards-based many years ago.

To the extent there is any standard now for the fork of DOCX that is actually in use in Microsoft Word, it is an internal, proprietary standard of Microsoft.

Every word processor that is not Microsoft Word has the problem that it is never completely successful in *exporting* into the DOCX format used by Microsoft Word. Only the simplest documents ever have fully successful exports into Microsoft Word.

For any word processor that is not Microsoft Word, there is no reason to have any confidence that an export into DOCX format that works today will work tomorrow. Microsoft changes ("extends") its DOCX format from time to time, without warning. Contributing to uncertainty and instability and professional liability risks, the USPTO also changes its own proprietary DOCX-to-PDF rendering engine from time to time, without warning.

Now let's return to USPTO's quest. USPTO said it was trying to figure out if there is an intermediate-storage word processor format that satisfies several requirements:

- the format is actually standards-based right now
- the standard is really an open standard
- every word processor is able to either save in that format or export into that format
- the saving or exporting is stable (likely to work the same way tomorrow that it works today)

As it turns out, there is such a format. The format is ODF -- OpenDocument Text. ODF is (try to guess from the name!) a standards-based format. Every word processor is able to save or export the ODF format. Everything about ODF is stable – there is no Microsoft making changes ("extensions") at undisclosed times. There are open-source rendering engines for rendering ODF into PDF.

If USPTO really feels that what it must do, going forward, is receive US patent applications using some word processor intermediate storage format, the correct answer is not DOCX, it is ODF.

As mentioned above, the Library of Congress has approved ODF format. See https://www.loc.gov/preservation/digital/formats/fdd/fdd000247.shtml . The Library of Congress says:

> As of 2020, office software suites using ODF as native file format include: LibreOffice, Collabora, Apache OpenOffice, and Calligra.

The Library of Congress enumerates several examples of governmental policy documents that mandate ODF among editable documents, including:

- The ***United Kingdom***. Sharing or collaborating with government documents, which mandates ODF 1.2. The UK government announced format choices in July 2014. See also Open standards for government, which is updated in place.
- In 2012, ***Portugal*** issued a regulation incorporating a list of mandatory formats. The only editable format for documents listed was ODF 1.1. See regulation in Portuguese and the list as presented in Computer Weekly.
- In 2009, ***Norway*** adopted a new set of obligatory information technology standards, mandating ODF as the only editable format for exchanging documents between the government and users by email. See announcement and summary in English.
- ***Brazil***'s ePING (Standards for Interoperability for Electronic Government) includes ODF 1.2 and ISO/IEC 26300: 2008 as the only editable formats for office documents. Several other South American nations appear to have similar regulations.

The European Commission recommends supporting OpenDocument format (https://joinup.ec.europa.eu/collection/open-source-observatory-osor/news/ec-recommends-supporting-open):

> All European institutes should be able to use the Open Document Format (ODF) in exchanges with citizens and national administrations, says Vice-President of the European Commission Maroš Šefčovič, in response to questions by member of the European Parliament Amelia Andersdotter. "There is no lock-in effect whatsoever, and no contradiction with the Commission's strategy on interoperability."

If the USPTO were to do the right thing and adopt ODF as the open-source format for filing of patent applications, it would find itself in good company.

The yearlong study. We now turn to the USPTO's mysterious "yearlong study" that supposedly concluded that no version of PDF can be used or trusted for filing of patent applications at the USPTO. Members of the patent filing community first heard about this mysterious "yearlong study" in 2020, when the USPTO published a Federal Register notice entitled *Setting and Adjusting Patent Fees during Fiscal Year 2020*, dated August 2, 2020 (85 FR 46932). This is the FR notice that communicates the USPTO's conclusion that if the USPTO is going to force applicants to change from what they were doing in the past, and in particular if the USPTO is going to force applicants henceforth to hand in some particular format for US patent applications, then those at the USPTO know what's best, and what's best is not some particular flavor of PDF. What's best (according to the USPTO) is Microsoft Word DOCX format.

The Federal Register notice said, in four places:

> The USPTO conducted a yearlong study of the feasibility of processing text in PDF documents. The results showed that searchable text data is available in some PDFs, but the order and accuracy of the content could not be preserved.

Practitioners tried to make sense of these two sentences. The impression that the authors of the FR Notice gave is that the yearlong study somehow worked out that there was no PDF variant that would serve the USPTO's needs, and thus that there was no choice but to force filers to use Microsoft Word (that is, DOCX).

The Acting Commissioner for Patents provided a copy of the "yearlong study" to me a few months ago, and I have reviewed it and have blogged about it. (A copy of the yearlong study is available at https://www.oplf.com/AEEC-AASET-Text2PTO-POC%20WhitePaper-v1.0%20FINAL.pdf .) As it turns out, the yearlong study tried to answer a very different question, a question that no one actually needed the answer to.

What the study tried to figure out was whether, in 2018, there existed some off-the-shelf commercial product that could take as its input a random sampling of the actual PDF files that filers had been filing at the USPTO, and could consistently extract usable text from most or all of those PDF files. Anyone who was even passingly familiar with PDF formats in 2018, and who had even passing familiarity with the word processors and PDF tools actually used by filers in 2018, could have provided the answer to this question to the USPTO for free. The answer was "of course not!" The authors of the study probably likewise knew perfectly well that the answer was "of course not!" but had no incentive to tell this to the USPTO before commencing the "yearlong study". Only by carrying out the yearlong study would the authors of the study be able to send out a bill and get paid.

It seems clear that USPTO carefully avoided conducting a good-faith study of PDF formats to see if any PDF format was in fact well suited to the USPTO's needs. It seems the USPTO

carefully avoided, in particular, ever learning of the existence of two PDF variants called *PDF/A Level A (accessible)* and *PDF/UA*. *PDF/A Level A (accessible)* is an ISO standard which, among other things:

- requires that all resources (images, graphics, typographic characters) must be embedded within the PDF/A document itself, and
- requires that text be extractable and the logical structure must match the natural reading order.

I was able to learn of these two PDF variants, and their usability for delivering characters to the USPTO, in a mere fifteen minutes of mouse clicking. The USPTO seems to have gone out of its way to carefully avoid ever doing the fifteen minutes of mouse clicking that would have been needed for the USPTO to learn that there is a PDF format that is ideally suited to USPTO's needs.

One of the members of the EFS-Web listserv (David Boundy) summarized USPTO's mistake in a succinct way that I will paraphrase here:

- The PDF standards such as *PDF/A Level A (accessible)* or *PDF/UA*, for good or for ill, are designed to solve the relevant problem of providing characters to someone like the USPTO.
- The Microsoft Word DOCX format is designed for editing and intermediate storage, not for interchange.

Understanding PDF/A Level A (accessible) and PDF/UA formats. Enormous amounts of time and energy have been spent on standards-setting work to define PDF formats that assure a blind person that he or she will be able to have the content of the PDF file read aloud. The effort has succeeded. Unlike the ill-fated DOCX standard-setting activity that came under Microsoft's control and then ceased to be a standard more than six years ago, the standard-setting work for PDF files that can be read aloud (that is, PDF files that are "accessible"), has not been controlled by any single industry player. This work has led, for example, to an international industry standard called ISO 14289-1:2014. This standard provides that the textual content is provided in "logical reading order". Tags permit the "reader" of the PDF to make perfect sense of headings, lists and tables. All fonts are embedded, and text is mapped to Unicode. Such PDF files are trusted by applicants. The USPTO ought to accept such PDF files as a way to provide characters to the USPTO.

This brings us back around to the all-important "yearlong study" which the USPTO carried out, and which supposedly led to the conclusion that PDF was not the right way to go for providing character-based information to the USPTO. We recall that in *four places*, the related Federal Register notice said

> The USPTO conducted a yearlong study of the feasibility of processing text in PDF documents. The results showed that searchable text data is available in some PDFs, but the order and accuracy of the content could not be preserved.

This is simply a lie, when these words are applied to real-life options for filing of patent applications. At the time the study was carried out, the "accessible" PDF formats were available, and they were industry standards, and they insured not only that searchable text data is available, but also that its "order" and accuracy were preserved.

USPTO's DOCX program is incompatible with USPTO's own rules about signing of inventor declarations. If a practitioner wishes to avoid getting dinged with the USPTO fee for handing in a signed inventor declaration later than filing day, then the USPTO rules require the practitioner to follow particular steps in a particular workflow. Basically the practitioner must do these things:

1. Prepare a patent application in a word processor;
2. print out the patent application to be a PDF file;
3. show the PDF file to the inventor;
4. obtain the inventor's signature on the inventor declaration; and
5. e-file the PDF file and the signed inventor declaration.

Note that steps 3 and 4 can often be done in a comfortable, leisurely way, at a time and place that works for the inventor.

This workflow is, of course, impossible within USPTO's DOCX program. To satisfy the USPTO's DOCX program, the practitioner must follow very different steps:

1. Prepare a patent application in a word processor;
2. export the document in one or another of the many variants of DOCX;
3. upload the DOCX file to the USPTO e-filing system;
4. find out in real time how the USPTO will modify the document for USPTO's systems;
5. capture this USPTO-modified version of the patent application as a PDF;
6. show the PDF file to the inventor;
7. obtain the inventor's signature on the inventor declaration; and
8. e-file the patent application, including the signed inventor declaration.

Note that steps 5, 6 and 7 are required to be done *at the time of filing the patent application.* Thus, for example, if the patent application is being filed at 11 PM on some particular day, then the only way to accomplish this filing path is to *force the inventor to stand by at 11PM and participate in the e-filing process at 11PM.*

Note that step 4 is based on which version of USPTO's DOCX validation engine is being used. Recall that in the Director's blog, she states with some pride that this validation engine is now up to "version 18". This reminds us that it will be of no use to do a validation process on a

Wednesday if the actual e-filing is planned for the subsequent Thursday. The USPTO might, after all, change to version 19 on that Thursday morning.

It will not do even to try to rely upon a *same-day* validation process. The validation performed on, say, the Thursday morning might make use of version 19 of USPTO's (proprietary, undocumented, black-box) DOCX validation engine, and then if the e-filing is actually carried out on Thursday evening, the USPTO might have slipped version 20 of its DOCX validation engine into service.

From all of this, it is clear that if USPTO wishes to be respectful to applicants and practitioners in this area of inventor declarations, then the USPTO simply must permit a PDF file to be the authoritative document for the filing of the patent application. Or to say this in a different way, USPTO's insistence on forcing the file to file a DOCX file, which is then subject to the vagaries of a moving-target validation engine, is profoundly disrespectful to applicants and practitioners in this area of inventor declarations.

Why are decisionmakers within the USPTO being so stubborn about all of this? From the point of view of patent practitioners who are external of the USPTO, it is baffling why the decisionmakers within the USPTO are being so stubborn about all of this. It is likewise baffling why USPTO people write so many false things in their web sites and customer training materials. What is the perceived upside to lying about there supposedly being a present-day "standard" for DOCX when the truth is that the standards are six to twelve years old and no present-day word processor follows those old standards? What is the perceived upside to the USPTO's *actions* making clear that the USPTO does not actually trust a DOCX-formatted document (given that what the USPTO trusts is only a PDF or modified DOCX file generated by USPTO's black-box engines), while stating to customers that it supposedly believes that "as an open standard format, DOCX offers a safe and stable basis for authoring and processing intellectual property documents"?

The anecdotal experience of practitioners in the filing community is that USPTO's (black-box) DOCX validation engine springs fewer surprises on the filer if the DOCX file was created using Microsoft Word. Filers who export DOCX files from other non-Microsoft word processors seem to encounter far more surprises with USPTO's (black-box) DOCX validation engine.

The patent practitioner community has been baffled by this stubbornness and disingenuousness by USPTO people. Why do they do it? There are four prevailing guesses about this.

- *Guess 1.* Maybe some years ago, some USPTO person signed a contract with a USPTO contractor that locked in some sweetheart price for printing of US patents, and the contract promises that the USPTO will provide (Microsoft-Word formatted) DOCX files to the contractor to get that sweetheart price.
- *Guess 2.* Maybe Microsoft somehow directly or indirectly influenced the thinking of the decisionmakers within the USPTO to set up its systems to "reward" USPTO customers

who purchase and use Microsoft Word and to “punish” USPTO customers who use other non-Microsoft word processors.

- *Guess 3.* Maybe the place where the USPTO got its DOCX validation engine was Microsoft.
- *Guess 4.* Maybe some high-up person (or group of people) within the USPTO stuck his or her neck out several years ago, and said “DOCX is the only and best way to go” even though that was not true, and in the years since then this person (or people) has been absolutely unwilling and unable to admit error.

How to fix USPTO’s mess? Assuming the USPTO really refuses to accept PDF, and assuming that the USPTO sticks with its idea that an intermediate-storage word processor format is the right way to go for patent applications, then the only defensible way to fix USPTO’s mess is to adopt OpenDocument Text (ODF) as the filing format. Only ODF satisfies USPTO’s stated requirements:

- the standard is really an open standard,
- every word processor is able to either save in that format or export into that format, and
- the saving or exporting is stable (likely to work the same way tomorrow that it works today).

DOCX does not satisfy the first or third conditions, and only roughly satisfies the second condition.

I have to assume that ODF is out of the question for the USPTO. Suppose for example that Guess 1 is the correct guess for USPTO’s stubbornness. If so, then the simple fact is that ODF is not the same thing as Microsoft Word’s variant of DOCX. If Guess 2 is the correct guess, then the adoption of ODF would be unacceptable because it fails to reward Microsoft Word purchasers and fails to punish those who use non-Microsoft word processors. If Guess 3 is the correct guess, then the adoption of ODF is unacceptable because the USPTO has already locked in its use of the Microsoft-provided validation engine. If Guess 4 is the correct guess, then once again adoption of ODF is out of the question because this would be admitting past error in picking DOCX.

So on the assumption that ODF is out of the question, what can possibly be done by the USPTO to clean up the mess, that is not hostile and offensive to the filing community?

As a first step, USPTO should suck it up and admit to the fool's errand of DOCX, given that there is no present-day standard for it, and should instead make use of a document format for which there is an actual published industry standard. ODF being assumed to be out of the question, then the one remaining choice is Accessible PDF.

Let's suppose that in response to this, the USPTO says "oh but we can't get absolutely everything we want from the word processor information that is embedded in the Accessible PDF". This is almost certainly not true, but let’s assume it to be true for sake of discussion. Then let the USPTO simply announce:

- Filers are required to e-file in Accessible PDF.
- If the filer hands in a PDF that fails to comply with the "Accessible PDF" published industry standard, the filer will have to pay a $400 penalty.
- The filer is invited to hand in, in addition to the PDF file, a word processor file with a file name ending in the four letters "DOCX" that the filer certifies to be content-identical given whatever word processor the filer used to author or export the word processor file. The filer gets to use any word processor they like. The only requirement is that in the word processor that they used, this is the DOCX file that the word processor generated or exported, and the PDF that they actually uploaded matches a PDF that was exported from the word processor that they used.
- If the filer hands in the word processor file as just described, they get a $3.15 filing fee reduction.
- Note that this gives the USPTO everything that it says it wants, and more.
- If the USPTO challenges the content-identicality of the DOCX file that the filer employed to get the $3.15 filing fee reduction, then it is up to the filer to say which word processor they used, and it is up to the filer to prove that that word processor exported both that PDF and that DOCX file.

I picked $3.15 because that is USPTO's admission of how much money it presently spends to do the OCR on an image-based PDF. Obviously no practitioner is going to spend the time to hand in the DOCX file to save a mere $3.15. But that is apparently the correct price for the USPTO to pay to avoid having to do the OCR. Of course the USPTO could voluntarily increase its reward for handing in the DOCX file to a bigger amount of money, let's say $400.

Given that no two word processors actually generate the same DOCX file content even for the same "what you see on the screen" (given that there is no industry standard), this will mean the USPTO will from time to time receive a DOCX file (say from my Libre Office or from somebody's Google Docs) that does not serve USPTO’s wishes as well as a Microsoft Word file would have. The USPTO will then be very annoyed that it paid $400 for this non-Microsoft DOCX file. USPTO will have buyer's remorse. But USPTO will not be heard to complain, because for five years now they have been telling anyone who will listen that "DOCX is a standard".

The PDF file will, of course, be the authoritative document for all later purposes. The USPTO will preserve this PDF file for as long as it maintains any official application file. This PDF file will then be available at litigation time if any question ever arises about what the applicant filed.

The USPTO can probably get everything that it really needs from the “accessible” content in the PDF file. But to the limited extent that the USPTO finds some real or imagined deficiency in the “accessible” content of the PDF, the USPTO could offer a suitable incentive to the filer to hand in the DOCX file that the USPTO seems to so desperately require. As I say, if $3.15 is not a big enough incentive, then the USPTO could name some higher price.

There are several very important things about the approach just described.

1. USPTO absolutely must permit filers to file PDF patent applications under circumstances where the PDF file "controls."
2. The "controlling" PDF file needs to be preserved by the USPTO intact, without modifications, for the life of the patent plus a statute of limitations period.
3. It will be acceptable to filers, I believe, for the USPTO to require that the PDF file be "accessible". This is an industry-standard term with clear and unambiguous meaning. All present-day word processors can generate "accessible" PDF files, either directly through native export function or by readily available add-ons.
4. If the USPTO were to choose to *require* each filer to provide not only a PDF but also a "DOCX" file, then the only requirement for the format of the DOCX file that the USPTO should be able to impose is that *some word processor* yielded that DOCX file and that PDF file. The filer gets to pick whatever word processor the filer wishes to use. The USPTO is not allowed to "reject" the DOCX file or require the user to review or "accept" any USPTO validation or modification of that DOCX file. The only permitted grounds for "rejecting" the DOCX file would be that the filer was unable to prove that the filer's word processor yielded both that DOCX file and PDF file.

Note that the USPTO should not be heard to make any objection to item 4. After all, if the USPTO is telling the truth about DOCX when the USPTO says that there is a present-day "DOCX standard", then that is the end of it. Some random word processor selected by the filer generated both that DOCX file and that PDF file. That is the end of it.

To emphasize this point, USPTO says, to anyone who will listen:

> DOCX is a safe and stable open source format supported by many popular word processing applications, including Microsoft Word 2007 and higher, Google Docs, Office Online, LibreOffice and Pages for Mac.

So if a filer files some PDF, and later hands in a file from Google Docs or Libre Office or Pages for Mac that ends in the letters "DOCX", and if the filer is able to show that Google Docs or Libre Office or Pages for Mac generated both that PDF and that DOCX file, then that should be the end of it. USPTO should accept that DOCX file and should not be permitted to impose any further requirement like reviewing or accepting any USPTO validation or modification of that DOCX file.

The fool's errand. We return now to the title of this essay -- *The Fool's Errand That Is DOCX*. The reward, such as it is, for the valiant reader who has somehow found a way to stay awake through this lengthy discussion, is that I will now explain what I mean by the title.

It might be thought that when I selected this title, perhaps I was trying to communicate that it is a fool's errand for the USPTO to have tried to make DOCX work for the filing of US patent applications. And it probably was and is a fool's errand that the USPTO tried to do this. But that

is not what I was trying to communicate in this title. I was trying to communicate my near-certain feeling that it is a fool's errand to try to use reason and logic to engage the USPTO in a meaningful discussion of how the USPTO could gain the cooperation of practitioners in obtaining e-filed character-based US patent applications.

When I see that the most prominent paragraph on the USPTO's DOCX web page is a paragraph containing three pants-on-fire false statements about DOCX, I despair of any hope of even reaching a shared understanding with the USPTO about what is true and what is false in the world of word processor formats and open standards.

When I see the USPTO setting forth its supposed criteria for selecting a word processor intermediate storage format that would work well (open standards, stable, nonproprietary), and the one correct answer is ODF, but the USPTO fails to get that answer, I despair. When I see the USPTO wrongly stating that the DOCX format supposedly satisfies all those criteria, when in fact it satisfies none of them, I despair.

When I see the USPTO claiming that it spent an entire year conducting a study that was supposedly directed to understanding PDF, and when the USPTO said the conclusion was that PDF could not be used … when text-rich versions do exist (*PDF/A Level A (accessible)* and *PDF/UA*) and I was able to find them in a mere fifteen minutes of clicking around in Google … I despair.

When I see the USPTO in its August 2, 2020 Federal Register notice, completely misrepresenting the findings of that "year-long study" … I despair.

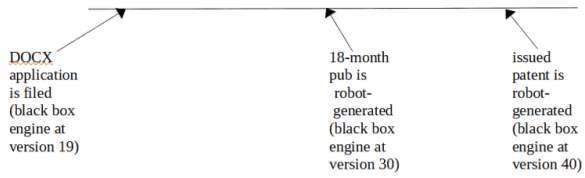
The fool's errand is my wish that I could use reason and logic in a meaningful discussion with the USPTO as to whether DOCX is the answer, or whether perhaps DOCX is not at all the answer, and maybe ODF or *PDF/A Level A (accessible)* or *PDF/UA* is the answer.

Exhibit B

JUNE 20, 2023 BY OPPEDAHL

One reason why USPTO's DOCX initiative presents professional liability risks

One of the many reasons why the USPTO's DOCX initiative presents professional liability risks arises from USPTO's use of an ever-changing "black box engine" to render DOCX files into PDF files. The black box engine is by now up to at least version 19.



click to enlarge

The timeline above offers a reminder of one of the ways that the ever-changing nature of the USPTO's black box DOCX-to-PDF rendering engine puts applicants and practitioners at risk.

We all know that DOCX is not a human-readable computer file format. The only way that you or I as a human being can "view" a DOCX file is *by opening it with some word processor another* and seeing how the content of the file gets rendered into human-readable form. This rendering might happen on the screen of the word processor. This rendering might happen when the word processor prints the content to a printer.

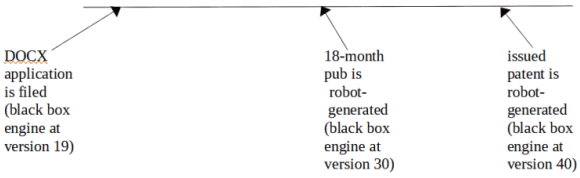
We all also know that it is commonplace to run into situations where three human beings, at different locations, opening the same DOCX file using three different word processors on the same day, can see three non-identical renderings on their computer screen or on their printers. Likewise the USPTO's own black box engine might render things differently on a particular day than some user's word processor. This illustrates the professional liability risks ***across different locations on some particular day***.

A moment's thought prompts a realization that the professional liability risks extend not only across locations on some particular day, ***but also extend over time***. If you were to open a particular DOCX file today, using today's version of (say) the Microsoft 365 cloud-based word processor, you might see one rendering on your screen. But if you were to open that same DOCX file a year from now (after some 365 days had passed), it is impossible for you to know for certain that the rendering on your screen would be identical. The software is cloud-based and would likely have changed from time to time over those 365 days.

Which brings us around to the USPTO's proprietary and ever-changing black box engine for rendering DOCX files into human-readable form. How do we know it is ever-changing? Director Kathi Vidal revealed this on her blog on December 19, 2022. She said:

> We've heard from some of you that you are concerned the validated DOCX version or the USPTO-generated PDF version may contain a discrepancy. Though we saw discrepancies in earlier versions of the tool, we considered your feedback and have updated the tool accordingly. It is now at a very advanced stage (***version 18***).

(emphasis added.) The engine changed again in March of 2023, bringing the version number up to at least 19. So let's look at a timeline. Let's suppose you file your DOCX patent application on a day when the USPTO's black-box rendering engine happens to be at version 19.



click to enlarge

This prompts a realization that the USPTO's black-box rendering engine might be at version 30 when the USPTO uses the engine to generate *the 18-month publication* of your DOCX patent application. And the USPTO's black-box rendering engine might be at version 40 when the USPTO uses the engine to generate *the issued patent* from your DOCX patent application.

This means that no matter how closely you try to proofread the rendering by the USPTO's engine on the day you file your DOCX patent application, this is of no help in predicting

whether some later version of the USPTO's rendering engine would render your DOC patent application.

This means that even if you look at the 18-month publication by the USPTO of your DOCX patent application, and maybe do not see any rendering problems, this this is of no help in predicting whether the later version of the USPTO's rendering engine in use *at the time of issuance* of your DOCX patent application might nonetheless introduce errors into your issued patent.

Exhibit C

JUNE 5, 2023 BY OPPEDAHL

Deficiencies in the "auxiliary PDF" approach for DOCX filing

There are many reasons why the USPTO's "auxiliary PDF" approach, which the USPTO hopes would induce reluctant DOCX filers to do DOCX filing, is unacceptable. (See USPTO blinks a second time on auxiliary PDF with DOCX filing.) Here are some of the reasons.

- The Federal Register notice carefully avoids saying that the uploaded PDF "controls" or is "authoritative".
- The USPTO does not come out and expressly promise that the message digest in the ack receipt *will match* the auxiliary PDF file.
- The USPTO does not even come out and expressly promise that *there will be a message digest* in the ack receipt for the auxiliary PDF file.
- Until now the USPTO has not been storing the auxiliary PDF bit-for-bit in SCORE. Instead, the auxiliary PDF gets dismembered into TIF images (one per PDF page). Then the TIF images get downsampled or upsampled from their original resolution to 300 dots per inch, thus losing resolution. The TIF images also get halftoned, which ensures blurring for any gray scale or color in the PDF. If and when the filer tries to download what purports to be the auxiliary PDF from IFW, the file downloaded from IFW it will be a different file size than the actual auxiliary PDF that the filer actually uploaded, and it will fail to match the message digest from the ack receipt, and it will be blurred, sometimes beyond readability.
- It is not at all clear from the Federal Register notice how the filer can escape the legal consequences of having clicked on the adhesion contract that says the D2 docx file (not the PDF) is the controlling document.

- The Federal Register notice carefully avoids saying how closely the auxiliary PDF has to "match" the D1 DOCX file (or the D2 DOCX file). Suppose they don't really match very closely, maybe because the USPTO's DOCX engine renders things strikingly differently than the filer's word processor does. Is the USPTO really committing to issuing a CofC to say that the issued patent is really what the auxiliary PDF said, no matter how much it diverges from the USPTO's rendering of the D1 or the D2?

NON-DOCX PENALTY

Dan Feigelson

JUNE 5, 2023 AT 1:57 PM

"The United States Patent and Trademark Office (USPTO) is continuing to modernize and streamline its patent application systems to support robust and reliable patent rights, speed the issuance of patents, and reduce the costs and barriers of global patent protection. The submission of patent applications in DOCX format facilitates the USPTO's ongoing efforts."

Sure. And the earth is flat.

JULY 1, 2023 AT 2:24 PM

Can anybody explain why the PTO goes so far out of their way to mangle the pdf files that applicants submit, as opposed to simply SAVING the damned things?

oppedahl

JULY 1, 2023 AT 2:52 PM

That's easy. I answered your question during one of my recent risks-of-DOCX webinars. When the USPTO decided that ePave was a failure, and then decided to start taking PDF patent applications, the USPTO tried to create its own software for this and failed. At that point USPTO licensed EPO's "Phoenix" software which is what EPO was using to manage its image-based patent filing system. This must have been terribly embarrassing for the USPTO from an institutional point of view because it was all to much like admitting that some other patent office was better at doing something than the USPTO was. Anyway, one thing that was baked into Phoenix at a very basic level was that it only really stored TIF images, one per page. So if the applicant were to e-file (say) a ten-page PDF, the first thing that would happen is the PDF getting dismembered into ten one-page TIF images. Later if the applicant wished to download a file from the (Phoenix renamed as EFS-Web) system as a PDF, the only way to do it was to stitch the TIF images back together on the fly, into a simulacrum of the original PDF.