IJDC | *General Article*

How Valid is your Validation? A Closer Look Behind the Curtain of JHOVE

Michelle Lindlar
Leibniz Information Centre for Science and Technology

Yvonne Tunnat
Leibniz Information Centre for Economics

Abstract

Validation is a key task of any preservation workflow and often JHOVE is the first tool of choice for characterizing and validating common file formats. Due to the tool's maturity and high adoption, decisions if a file is indeed fit for long-term availability are often made based on JHOVE output. But can we trust a tool simply based on its wide adoption and maturity by age? How does JHOVE determine the validity and well-formedness of a file? Does a module really support all versions of a file format family? How much of the file formats' standards do we need to know and understand in order to interpret the output correctly? Are there options to verify JHOVE-based decisions within preservation workflows? While the software has been a long-standing favourite within the digital curation domain for many years, a recent look at JHOVE as a vital decision supporting tool is currently missing. This paper presents a practice report which aims to close this gap.

Received 20 October 2016 ~ *Revision Received* 23 February 2017 ~ *Accepted* 23 February 2017

Correspondence should be addressed to Michelle Lindlar, Welfengarten 1B 30168 Hannover Germany. Email: michelle.lindlar@tib.eu; Yvonne Tunnat, Düsternbrooker Weg 120, 24105 Kiel. Email: y.tunnat@zbw.eu

An earlier version of this paper was presented at the 12th International Digital Curation Conference.

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Introduction

The results of the 2015 OPF community survey lists JHOVE, alongside DROID, as the most important workflow tool in digital curation and preservation environments (OPF, 2015a). JHOVE, which derives its name from its 2005[1] origin as the "JSTOR/Harvard Object Validation Environment", was designed as a flexible and extensible framework in which modules support different file formats. The software may be used as a stand-alone tool, but it is also frequently embedded in preservation systems such as Preservica, Archivematica or Rosetta, as well as in extended repository solutions for digital preservation or research data management.

The wide-spread use of the tool may lead especially inexperienced users to follow the tool's output blindly. But is JHOVE indeed the authoritative voice on file format validation? Can we trust the output? Do we know and understand what it is based on?

JHOVE is a modular tool with a framework layer for generic tasks and a module layer for the actual file format analysis. As this paper deals with the validation aspect in regards to specific formats, the focus is on the module layer. For the scope of this paper three of the 15 different format modules[2] were chosen: PDF, TIFF and JPEG. The reasoning behind choosing these three modules is presented in the Background section of this paper, which will also include a brief discussion of what constitutes a digital object's 'well-formed' and 'valid' status. The Methodology section will describe the criteria used against the three modules in their respective evaluation sections. The evaluations of the three modules is presented in separate chapters, describing the status-quo as well as current work being undertaken by the digital preservation community to improve the trust in and usability of the JHOVE modules. The paper concludes with a brief conclusion and outlook.

Background

Format Selection

The file formats PDF, TIFF and JPEG and their respective JHOVE modules were chosen based on the criteria 'usage of format', 'complexity of format', 'complexity of module'. The authors set out to evaluate three modules that validate popular file formats which differ in file format complexity, subsequently leading to different complexity on the JHOVE module layer as well.

The wide usage of the PDF format is of course not limited to digital archives. Duff Johnson's 2014 study on popular document formats on the web showed that 77% of the returned hits were PDF (Johnson, 2014). The TIFF format, on the other hand, still is the most widely used preservation master format in digital archives, depending on the study between 87 to 94% use TIFF as their digital preservation master format (Wheatley et al.,

1 The first production release of JHOVE was dated 2005-05-26. See: https://web.archive.org/web/20060118221635/http://hul.harvard.edu/jhove/news.html

2 One module may support different versions of a format. At the time of writing this paper, the following modules were available: AIFF, ASCII, GIF, GZIP, HTML, JPEG, JPEG2000, PDF, PNG, TIFF, UTF-8, WARC, WAVE, XML and BYTESTREAM.

2015). As for the JPEG format, it remains a widely supported format on a global scale, entering digital archives through various workflows (Library of Congress, 2013).

The formats differ significantly in complexity, with the PDF file format family being the most complex, making validation a challenging task. This also reflects in the JHOVE module with sometimes misleading error output, posing a potential risk when basing preservation decisions solely on the verdict of JHOVE. Recently discovered misinterpretations of the standard in the module, such as in the case of CrossRefStream values, have been leading to false negatives[3]. TIFF, on the other hand, is a clearly defined standard, resulting in a JHOVE validation module with comprehensible findings and reliable output for preservation purposes. Due to the format's wide adoption and stability, as well as to the straightforwardness of the standard, other tools such as the validators LibTIFF[4] or DPF Manager[5] exist and can be used to verify JHOVE module output. While the JPEG file format standard ranges between PDF and TIFF when it comes to the format's complexity, the JHOVE-module is an example for a low-level validation module – it has not been updated since 2007 and only validates against 11 criteria, most of them being header values for the different JPEG formats covered. The question at hand is if this really is sufficient validation for digital preservation purposes. Other tools such as Bad Peggy[6] exist to validate specific JPEG file format family parts and can be used instead of, or in addition to JHOVE.

Table 1 reflects the standards' and modules' complexity in number of pages in specification for the respective file format and number of possible JHOVE validation errors for the respective module

Table 1. Overview analysed formats.

	Number of pages in specification	Number of possible JHOVE errors
PDF	1310	152
JPG	481[7]	13
TIFF	121	68

Definitions of Well-Formed and Valid

File format validation processes analyse whether a digital object adheres to the specification of the format it claims to be. Validation results are usually broken down into two different conformance levels: *well-formed* and *valid*. An XML file, for example, is considered to be well-formed when it meets a fixed set of criteria as defined in the W3C Extensible Markup Language Standard document[8]. While well-formed XML objects comply with the XML specification, valid XML objects comply with an XML schema. In short, well-formedness addresses the syntactic correctness while validity describes the semantic correctness of an object's conformity to the file format it purports to be.

3 See Github bug report for JHOVE: https://github.com/openpreserve/jhove/pull/97
4 LibTiff: http://libtiff.org/
5 DPF Manager: http://dpfmanager.org/index.html
6 Coderslagoon: Bad Peggy 2.1: https://www.coderslagoon.com/#/product/badpeggy
7 Without amendments and references (ISO/IEC 10918). See: https://jpeg.org/jpeg/index.html
8 To give an example: to be well-formed, all elements within an XML must be delimited by start and end tags

While plain prescriptions of well-formedness and validity would be desirable within any standard documentation, the information is not always easy to find and often even ambiguous. An example for this is the PDF standard, where the requirements for well-formed objects are to be described in chapter 3.4 on file structure. However, this chapter also introduces characteristics which are optional, such as the newly introduced object streams – along with required dictionary values if the object is contained within the file. Unfortunately, this makes it very hard to derive exact requirements from the specification (Adobe, 2004).

The JHOVE modules contain clear descriptions of how well-formedness and validity of the file format is defined in the context of the module and what characteristics of the digital object's file format are checked.

The term 'validation' may in itself become ambiguous in a curational context. While JHOVE extracts technical metadata which may allow checking the digital objects' compliance to institutional policies (e.g., only uncompressed TIFF), it is not a policy checker. To better differentiate between these different concepts of 'validity', the PREFORMA project, which has put forth the software DPF manager, veraPDF and MediaConch, calls these tools 'conformance checkers', differentiating between 'implementation checker' (i.e., the standard) and 'policy checker' (i.e. institutional requirements) layers (PREFORMA, 2015).

Methodology

We set out evaluating JHOVE from a practitioner point of view by taking one step back and asking ourselves what we actually expect from a validation tool. The criteria were then grouped into categories and assigned to either the framework or the module layer of the software. The result of this process is shown in Figure 1. The categories pertaining to the module level are briefly explained below.

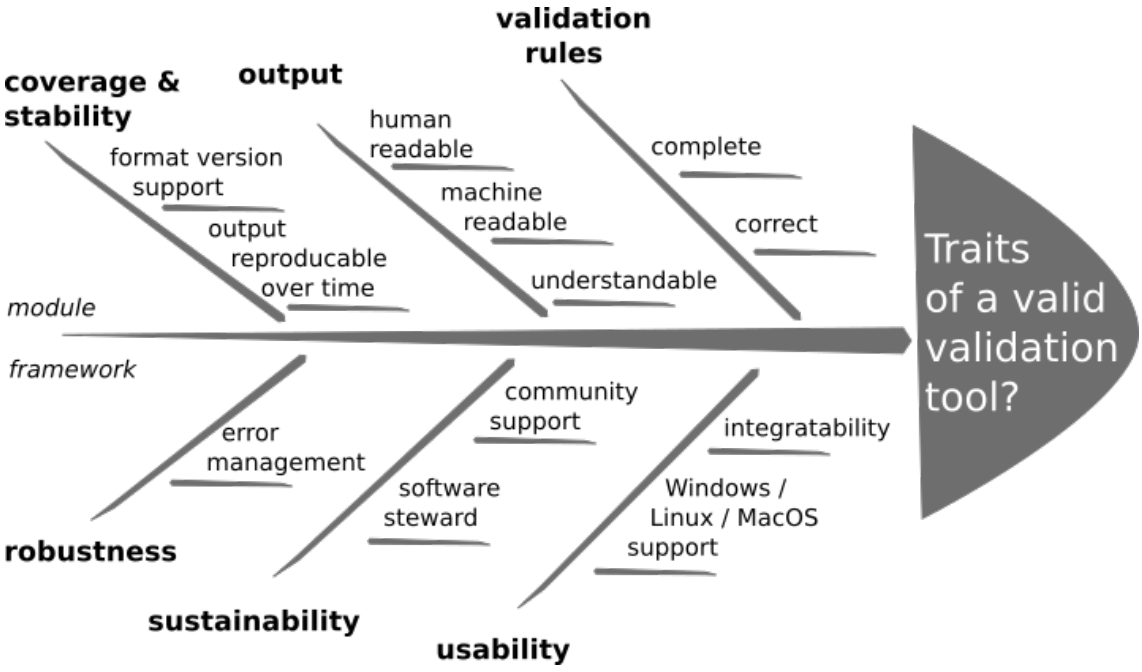

Figure 1. Traits of a valid validation tool – grouped into module and framework layers.

Coverage and Stability

In general, two aspects about a module's degree of coverage are relevant: how much of the file format family, meaning different versions, is covered by the module and how reproducible the validation results are over the course of time, i.e. throughout different module versions. The answer to the latter is a generic one for JHOVE. Since 2013, the software has been located on Github, allowing code changes to be tracked via versioning. Prior to that, JHOVE was made available via Sourceforge, where the code revision history is still available[9]. The versioning of the software allows a view of different software versions over time, as changes to code which may have an effect on validation outcome can be tracked. Furthermore, the OPF conducts an automatic regression testing routine for JHOVE when a new version is released. [10] For this, the modules are run against a fixed test-corpus and the outcomes compared to those of the previous version (OPF, 2015b). Being under the stewardship of the OPF, JHOVE development and maintenance is being monitored closely and regulated by a product board.[11] The degree to which a module covers a file format family differs from module to module and is covered in the respective module evaluation sections.

Output

The output of the validation tool describes whether an object adheres to the file format standard and is indeed well-formed and valid. If the object is not well-formed and valid, error messages shall exist, informing the user of where and how parts of the file violated the file format standard rules. As validation tools are often integrated into a larger workflow solution, the output should be both human and machine-readable. This is the case for all JHOVE modules. Ideally, the error message is furthermore intelligible for the decision maker who analyses the errors, and, if applicable, includes a workflow suggesting how to deal with the error. If there is no such thing and the error message cannot be understood, there is nothing left to do but to trust the tool and reject the file. This is why the transparency and intelligibility is an important factor for the evaluation of a JHOVE module.

Validation Rules

The validation rules used by the modules to check a file's conformance must be correct and complete. There are two possible erroneous deviations: If a tool marks a file which does not adhere to the specifications regulation as valid, this is a false positive. If on the other hand, the tool marks a file which complies with the standard as invalid, this is a false negative.

There are different ways in which the correctness of validation rules can be evaluated. One method is to knowingly create files which either conform to or violate certain aspects of the standard. This, of course, requires a solid understanding of the standards. A second method is to rely on rendering software, following the assumption that it was developed to interpret the file formats as described in their respective standards. However, this is not necessarily the case, as viewers are often tolerant and

9 Sourceforge revision history: http://jhove.cvs.sourceforge.net/viewvc/jhove/jhove/

10 The test-corpus and the scripts used are also available on Github:
https://github.com/openpreserve/jhove/tree/integration/test-root/corpora
https://github.com/openpreserve/jhove/tree/integration/jhove-bbt/scripts

11 JHOVE OPF product page: http://openpreservation.org/technology/products/jhove/

display files which violate the specification. A third method is to compare the output of one validation tool against the output of other validation tools. The problem here is that not all tools have alternatives to JHOVE (as in the case of PDF), most likely due to the high complexity of the format.

Within the scope of this paper all three methods were used. However, the highest focus was placed on the comparison of the validation output to that of other tools.

Module Evaluation – PDF Module

Curator/Digital Preservationist and the JHOVE PDF-module seem to embody the biggest love-hate relationship in the community. The module is known to be buggy[12], yet everyone relies on it.

A described earlier, PDF is a complex format. Additionally, due to the existing number of different format versions and profiles, which in return are based on versions that are again versioned in themselves[13], the lines between the file format family, the file format version and the profile seem to become blurry. In light of this, it is crucial to understand what the PDF-module checks against.

We are aware that PDF/A is the go-to-format if it's about long-term availability. However, JHOVE is not a tool we would recommend for PDF/A validation. The JHOVE PDF module was built for standard-PDF and the PDF/A profile check was implemented only as an additional feature. Usually JHOVE does not identify PDF/A files correctly and only runs a test against the PDF standard (Friese, 2014).

Alternative Tools

Currently, there are no alternative tools to check the validity of a standard-PDF file. However, there are a number of tools or tool suites which help us in further examining PDFs. The best known example is most likely Adobe's Preflight, a structure explorer and profile checker released with Adobe Reader Professional. An example for a freely available tool is the xpdf[14] suite, whose pdfinfo command can extract basic information such as the number of pages, the PDF version or if the PDF is tagged or encrypted. Another helpful command from the xpdf suite is pdffonts, which returns all fonts used in a file, including information such as the object number they are used in and whether they are embedded or not. Another excellent resource for troubleshooting problematic PDFs is PDFtk[15], in particular the tool suite included in the command-line PDFtk Server package. PDFtk allows one to decompress encrypted streams, to extract embedded metadata and to re-write the cross-reference dictionary.

A myriad of further PDF tools exist, some are wrapping a lot of analysis features, others are handling very specific tasks, such as printing the offsets to standard out.[16] While these tools allow one to manipulate and analyse the file format, none of them

12 See comment from original JHOVE developer Gary McGath on 2014-07-10: *"The PDF module has a history of bugs relating to page trees, […]. If other software doesn't complain, I'd be inclined to call this a JHOVE bug."* https://sourceforge.net/p/jhove/discussion/797887/thread/2050dc83/

13 For example: PDF/X-1a:2001 and PDF/X-3:2002 are based on PDF 1.3, PDF/X-1a:2003 and PDF/X-3:2003 are based on PDF 1.4, PDF/X-4 and PDF/X-5 are based on PDF 1.6.

14 XPDF: http://www.foolabs.com/xpdf/

15 PDFtk: https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

16 See: http://khkonsulting.com/2013/01/the-trouble-with-the-xref-table/

tackle a conformance checking against the file format's standard. This might very well be due to the almost overwhelming complexity and flexibility of the file format.

Due to this, unfortunately an extended comparison of the JHOVE PDF-Module against other validation software was not possible.

Coverage and Stability

According to the documentation[17], the PDF-module covers the following PDF versions and profiles: PDF versions 1.0-1.6, PDF/X-1 (ISO 15930-1:2001), PDF/X-1a (ISO 15930-4:2003), PDF/X-2 (ISO 15930-5:2003), PDF/X-3 (ISO 15930-6:2003), Linearized and Tagged PDF (available since PDF 1.4) and PDF/A-1 (ISO/DIS 19005-1). This list shows a few obvious gaps: PDF 1.7, which later become ISO 32000-1:2008, is not supported. Due to this, files that use features specific to PDF 1.7 or later may be reported as not well-formed or not valid. The same holds true for the lacking support for PDF/X-5 (ISO 15930-8:2010). Also, profiles such as the universal accessibility supporting PDF/UA (ISO 14289-1:2012-07) profile or the in the engineering domain the popular PDF/E (ISO 24517-1:2008) profile are not covered[18].

The PDF-module is currently in version 1.7, release date 2012-08-12 as per the module information page.

Output

PDF-module error messages are clear but not concise enough. Information such as "Invalid page tree node" or "Invalid structure attribute" could benefit from some additional information, such as which page tree node and why, or which structure attribute. Due to this, it is not easy to tell if the error has an impact on the long-term-availability of the file or to which paragraph in the PDF standards it refers to, making looking for a cure even harder (OPF, 2016a). Troubleshooting PDFs which failed validation almost always requires quite a bit of further analysis.

Currently, the OPF is working on more intelligible explanations for the JHOVE errors[19], taking a first step towards better understanding of the impact. Eventually, this will enable the curator to fix the problem, resulting in a valid and well-formed PDF, if desired.

Validation Rules

The PDF-module considers a file to be well-formed if it meets the basic syntactical requirements regarding header, body, cross-reference table, trailer and end-of-file marker.[20] Specifics are only described in regards to beginning- and end-of-file marker and in regards to the trailer, which must include the cross-reference table size and an indirect reference to the document catalogue. In regards to objects, the documentation states that they must be "well-formed". Validity criteria are divided into general validity for PDF 1.0-1.6 as well as some profile-specific validity criteria. The software information page gives the additional information that the module does not validate data within the content streams or encrypted data.

17 See: http://jhove.sourceforge.net/pdf-hul.html
18 JHOVE PDF Module: http://jhove.openpreservation.org/modules/pdf/
19 See: http://wiki.opf-labs.org/display/Documents/JHOVE+issues+and+error+messages
20 See: http://jhove.openpreservation.org/modules/pdf/

Comparing the profile specific validation rules against the profile references it becomes clear that JHOVE cannot be a definite validator for all profiles covered, as it has already been proven for PDF/A (Friese, 2014). Instead, the majority of the rules check towards generic PDF requirements.

As alternative tools do not allow a direct comparison of validation results, false positives and false negatives were analysed using existing and manually built examples.[21]

A common error in JHOVE is that of an 'Improperly constructed page tree'. This error is thrown if the PDF presents pages in page tree nodes which are not balanced, i.e. the page tree nodes do not contain an even spread of referenced page objects or page tree (sub-)nodes. Although balanced page tree nodes result in a better performance of viewers, it is not a requirement of the specification, but instead a concept introduced by the Acrobat Distiller Program (Adobe, 2006). As JHOVE reports it to be an error and the file to be invalid, this is an example of a false positive.

A false negative example can be easily reconstructed via the rotation property. While the rotation property of a page is optional, if present its value must be per standard a multiple of 90 (Adobe, 2004). However, JHOVE validates the file as 'well-formed and valid' regardless of the rotation property being 0, 90, 67 or 3. Viewers tested[22] neglected the invalid value and instead displayed at the default value (0, no rotation).

Conclusion for the JHOVE Module

PDF is a mighty format family with several thousands of specification pages. This makes the validation process especially cumbersome. The JHOVE PDF-module gives us a good starting-set of common denominators for validation criteria across different profiles. The syntactical well-formed criteria are crucial for the sustainability of the digital object, as basic errors such as missing end of file markers or missing document catalogue entries leave the file unrenderable. The PDF-module seems well-suited to spot such basic syntactical errors and also to hit 'high level' marks of profile validation. However, the user needs to be aware of the fact that the module is not suited for a complete profile check.

The bad news is that the error messages are hard to interpret, require a good amount of file format knowledge and that the module is known to be buggy with quite a few false negatives. Thankfully, the community is currently undertaking efforts to address this gap. However, it will take time and patience.

When using the PDF-module it is very important to understand the limitations of the module. An institution may choose, for example, to only focus on errors on the 'well-formed' level in a first step and to address the 'valid' errors at a later point in time when better validation rules are available.

21 The authors extended this work significantly in later research, leading to the publication of a PDF test corpus (Lindlar, Tunnat and Wilson, 2017a) and an accompanying publication (Lindlar, Tunnat and Wilson, 2017b), exploring JHOVE PDF module validation checks against a synthetic testset

22 Rendering was tested using three viewers: Acrobat Pro 11.0.15., Evidence 2.32, GSView 5.0

TIFF Module

The TIFF module[23] considers a file to be at least well-formed, if nine criteria – mostly dealing with the file header and the IFD (image directory file) – are met. Validity is a little bit more complex, looking for specific tags, and checking against valid formats and ranges of values.

Alternative Tools

There are many alternatives to check the validity of TIFF files. As the TIFF standard is pretty straightforward and easy to understand, it is not complicated to build a TIFF checker for one's needs, and many have done so, e.g. the SLUB Dresden with checkit_tiff[24].

The DPF manager is an open source conformance checker which checks if a TIFF file follows its specification. If it does not, it is marked as invalid, one (or more) error messages are provided and for each error the concerned page in the TIFF specification is referenced. Furthermore, it is possible to validate baseline TIFF, extended TIFF, TI/A and to create your own policies for validation. For this analysis, version 3.1 was used[25].

Validation is only a byproduct of ImageMagick[26], as it focuses on image creation, conversion and editing. It is a command line tool, although a basic GUI for the display of images is provided. For this analysis, version 7.0.3 was used.

ExifTool[27] is primarily meant for metadata extraction. It also gathers warnings and errors of an image file and can therefore be used for a superficial analysis of the quality of images as well. For this analysis, version 10.37 was used.

LibTiff[28] runs on UNIX and is able to give some information about the quality and the validity of a TIFF file. For this analysis, version 4.0.7 was used.

Coverage and Stability

The module supports the three major versions 4.0, 5.0 and 6.0 as well as standardized extensions such as TIFF/IT, TIFF/EP or GeoTIFF 1.0[29].

Output

JHOVE error messages for the TIFF module are almost generally understandable with a minimal level of file format knowledge (OPF, 2016b).

Validation Rules

An analysis run against 166 TIFF files from the Google Imagetestsuite led to the detection of several false positives (Tunnat, 2017a). While false negatives are more difficult to prove, the same analysis has shown at least two instances of false positive hits for the JHOVE TIFF module against the Google Imagetestsuite (Tunnat, 2017b).

23 JHOVE TIFF Module: http://jhove.openpreservation.org/modules/tiff/
24 checkit_tiff: https://github.com/SLUB-digitalpreservation/fixit_tiff
25 PREFORMA DPF manager: http://www.preforma-project.eu/dpf-manager.html
26 ImageMagick: http://coptr.digipres.org/ImageMagick
27 ExifTool: http://coptr.digipres.org/ExifTool
28 LibTIFF – TIFF Library and Utilities: http://www.libtiff.org/
29 JHOVE TIFF Module: http://jhove.openpreservation.org/modules/tiff/

Conclusion for the JHOVE module

The only real alternative for JHOVE for TIFF is the DPF manager, which has been developed only recently. All the other tools tested are for special needs (Baseline TIFF testing for checkit_tiff) or validation is only a byproduct, as for ExifTool and ImageMagick. LibTIFF might be an alternative tool, but it does not run on Windows and is therefore not easy to embed for Windows users.

As JHOVE is not free from false positives and most likely not free from false negatives as well, it probably should not have the last word in archiving decisions. Nevertheless, it is a decent and good working tool and there are no objections against using it in digital preservation workflows for a first orientation about the quality of the data.

JPEG Module

The criteria JHOVE tests against to determine if a file is well-formed and valid are clearly documented: three criteria for well-formedness and five for validity.[30] It is no surprise that the JHOVE JPEG module consists of 13 possible error messages only. The validation tool Bad Peggy can distinguish between at least 30 different JPEG errors. In a practical test, 28 different Bad Peggy error messages were found and only eight error messages of the JHOVE JPEG module (Tunnat, 2016a).

Alternative Tools

There are some tools out there which are able to check the validity of JPEG files. In this paper, we will focus on Bad Peggy[31], which is able to validate images like JPEG, PNG and GIF and detects damages. It enables the user to find broken files quickly and uses the java IO library to do so. It is also integrated in KOST-Val, the validation tool of the Swiss KOST. Furthermore, ImageMagick and ExifTool were used, which have already been described earlier in this paper.

Coverage and Stability

The JHOVE JPEG module supports JPEG (ISO/IEC 10918-1:1994), JFIF 1.02 (JPEG File Interchange Format), Exif 2.0, 2.1 (JEIDA-49-1998) 2.1, and 2.2 (JEITA CP-3451), SPIFF (ISO/IEC 10918-3:1997), JTIP (ISO/IEC 10918-3:1997) and JPEG-LS (ISO/IEC 14495).

Output

Some of the errors are easy to understand (Example: ‘Unexpected end of file’; some are not (Example: ‘Marker not valid in context’).

30 JHOVE JPEG module: http://jhove.openpreservation.org/modules/jpeg/

31 Community Owned digital Preservation Tool Registry (COPTR) – Bad Peggy: http://coptr.digipres.org/Bad_Peggy

Validation Rules

To test the reliability of the JHOVE JPEG module, the 98 files of the Google Imagetestsuite[32] were validated with the alternative tools Bad Peggy, ImageMagick and ExifTool. The results were captured in a spreadsheet (Tunnat, 2017b), sorted by different criteria. First, all the images that could be rendered by viewers like Paint and Windows Photos, second, all images which can be displayed, but somehow look bogus (chunks missing, all grey etc.) and lastly, all files which cannot be rendered at all. Analysing the output it is likely that JHOVE has one false positive and seven false negatives in the sample of 98 JPEG files (Tunnat, 2016c).

The seven false negatives are more worrisome. Bad Peggy and ImageMagick both agree that there is something wrong here. Besides, none of the seven files can be rendered by a viewer. One would really want some error output from JHOVE here, as these files obviously are corrupt. Furthermore, JHOVE misses seven out of 18 corrupt JPEG images in the analysis done for a blog post of the Open Preservation Foundation (Tunnat, 2016a). Those JPEGs could be opened in a viewer, but all bear some visually obvious errors. Bad Peggy, on the other hand, was able to detect all 18 corrupt images.

Conclusion for the JHOVE Module

The JHOVE JPEG module clearly misses quite a few corrupt images, as at least two tests have shown, one on the Google Imagetestsuite and one in an OPF Blogpost (see above). There is an upside, though, as Bad Peggy is an alternative tool which does not seem to miss out any of these files, although an infallibility of Bad Peggy cannot be tested within the scope of this paper.

Conclusion

The technological change not only applies to the materials we care for, but also to the tools we use within our workflows. This should be reason enough to regularly re-evaluate the tools we embed and base our preservation decisions on. As far as JHOVE is concerned, we know that it is widely adopted, embedded in many digital preservation workflows, and has been around for over ten years. However, large parts of the code have not been updated in quite a while. While many of us rely on the PDF-module, it still does not support PDF 1.7 – a version which was released nine years ago, in 2008.

While this paper only looked at three JHOVE modules, a recent OPF blog post by Johan van der Knijf indicated shortcomings and false positive/negative hits for the WAVE module as well (van der Knijf, 2016).

Both authors will continue to use JHOVE in their preservation workflows. However, the analysis put forth in this paper shows that supporting measures should be taken. If there is an alternative tool available for the file format – as in the case of DPF manager for TIFF or Bad Peggy for JPEG – it is recommendable to embed both tools in the workflow and to compare the results. Preservation watch activities should include constant awareness of new validation tools for the file formats in our archives.

The analysis has further underlined how crucial community work towards validation processes is. The JHOVE framework remains robust – much more robust than the validation modules themselves, which have been in place for years – but only now are

32 Google Imagetestsuite: https://code.google.com/archive/p/imagetestsuite/downloads

validation bugs becoming known. The community needs to join efforts in developing new validation rules and in checking existing ones against the standard. Recent activities lead by the OPF, such as the JHOVE hack day, the Document Interest Group's list of error messages, or the JHOVE product board have been a great start.

At the end of the day, valid validation can only be achieved if you understand the processes behind it and evaluate them regularly.

References

Adobe. (2004). *PDF reference: Fifth edition*. Adobe Portable Document Format, Version 1.6. November 2004. Retrieved from http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf

Adobe. (2006). *PDF reference: Sixth edition*. Adobe Portable Document Format, Version 1.7. November 2006. Retrieved from http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf

Friese, Y. (2014). *Ensuring long-term access: PDF validation with JHOVE? PDF association.* Retrieved from https://www.pdfa.org/ensuring-long-term-access-pdf-validation-with-jhove/

Johnson, D. (2014). *The 8 most popular document formats on the web.* Retrieved from http://duff-johnson.com/2014/02/17/the-8-most-popular-document-formats-on-the-web/

Library of Congress. (2013). *JPEG image encoding family*. Retrieved from http://www.digitalpreservation.gov/formats/fdd/fdd000017.shtml

Lindlar, M., Tunnat, Y., & Wilson, C. (2017a). Synthetic PDF Testset for File Format Validation. Dataset retrieved from: https://doi.org/10.22000/53

Lindlar, M., Tunnat, Y., & Wilson, C. (2017b). A PDF test-set for well-formedness validation in JHOVE: The good, the bad and the ugly. In *iPRE2017 Proceedings of the 14th International Conference on Digital Preservation, Kyoto, Japan.* Retrieved from https://doi.org/10.5281/zenodo.1228650

OPF. (2015a). *Open preservation community survey*. Retrieved from http://www.openpreservation.org/documents/public/OPFDigitalPreservationCommunitySurvey2015.pdf

OPF. (2015b). *JHOVE evaluation and stabilisation plan*. Retrieved from http://openpreservation.org/public/OPF_JhoveEvaluationStabilisationPlan.pdf

OPF. (2016a). *PDF errors in JHOVE*. List compiled during OPF JHOVE Hackday. Retrieved from http://docs.google.com/spreadsheets/d/1zyg4eqH6akoehI10fNhzroaW3CgcwUjkNlvvVZvFDyo

OPF. (2016b). *TIFF errors in JHOVE*. List compiled during OPF JHOVE Hack Day. Retrieved from https://docs.google.com/spreadsheets/d/1zyg4eqH6akoehI10fNhzroaW3CgcwUjkNlvvVZvFDyo/

PREFORMA. (2015). *PREservation FORMAts for culture information/e-archives*. Retrieved from http://www.digitalmeetsculture.net/wp-content/uploads/2015/04/PREFORMA-General-Presentation-v2.pdf

Tunnat, Y. (2016a). *Error detection of JPEG files with JHOVE and Bad Peggy – so who's the real Sherlock Holmes here?* Retrieved from http://openpreservation.org/blog/2016/11/29/jpegvalidation/

Tunnat, Y. (2016b). *Google JPEG ImageTestSuite validation tool comparison: Comparison of all tools*. Retrieved from https://docs.google.com/spreadsheets/d/1v0JbJZFs_4Oy405li_xkmxAoWhAqQs-nB9yeNAg3tnI/edit#gid=0

Tunnat, Y. (2016c). *Google JPEG ImageTestSuite validation tool comparison: JHOVE false findings*. Retrieved from https://docs.google.com/spreadsheets/d/1v0JbJZFs_4Oy405li_xkmxAoWhAqQs-nB9yeNAg3tnI/edit#gid=198401246

Tunnat, Y. (2017a). *TIFF format validation: Easy peasy?* Retrieved from http://openpreservation.org/blog/2017/01/17/tiff-format-validation-easy-peasy/

Tunnat, Y. (2017b). *Google TIFF ImageTestSuite validation tool comparison*. Retrieved from https://docs.google.com/spreadsheets/d/1AsJNXEjJlfYau1JmCj7YiEr1WNNlA-hHvXoJ4O3A7nw/edit#gid=245183935

van der Knijff, J. (2016). *Breaking waves (and some FLACs).* Retrieved from http://openpreservation.org/blog/2017/01/04/breaking-waves-and-some-flacs/

Wheatley, P., May, P., Pennock, M., Kimura, A., Whibley, S. (2015). *TIFF format preservation assessment. Chapter 2.2.* Retrieved from http://wiki.dpconline.org/images/6/64/TIFF_Assessment_v1.3.pdf