Digital Preservation Stage Boss One

By Ross Spencer

With thanks to Richard Lehane (State Records NSW), Becky McGuinness and Carl Wilson (Open Preservation Foundation), and the PRONOM team (The National Archives, Kew, UK)

At Archives New Zealand we were finding 'WAVE' files becoming a bottleneck of one of our ingest processes. The result initially looked odd to me where I had thought I had understood in the past that format identification would not take longer to divine than a checksum. My rationale being that to identify a file, DROID, or equivalent tool, would rarely need to read the whole file, rather, it should be able to read signatures from offsets relative to the beginning, or end of file.

On top of that, I assumed that even if the identification tool was 'slow' then it would only take as long as checksum generation, never longer. My rationale for this being that reading from disk is the slowest part of the process – once the file's contents were in memory, the tool working with the file would continue its computations in-memory.

It seems that the word 'rarely' may have been the error in my original assumptions; and finding those challenged I was eager to create an empirical picture of the performance of format identification vs. checksum generation and created a handful of experiments to do this.

The work contributing to these experiments can be found in my GitHub repository Digital Preservation Stage Boss One.

Summary of the Experiments

The same processes were run across three corpora, creating three tests in total:

1. Measure performance across the Govdocs Selected Corpus

   This is our first benchmark. Publicly available so that these experiments can be verified by others, the corpus represents a set of files that we might migrate into a government digital archive.

2. Measure performance across a simulant of the Govdocs Selected Corpus.

The purpose of this test was to understand the behaviour of an identification tool on a set of files that should not match a signature specification in PRONOM. The Govdocs simulant is created from a '***du***' manifest of the file sizes in Govdocs Selected corpus. Each size is read and a file of equal size, populated with random data is created. The result is 26,124 files, totalling 31.4GB in size. Anyone can create a set for themselves with the following code: https://github.com/exponential-decay/digital-preservation-stage-boss-one/tree/master/fakegovdocs-filegen

3. Measure performance across a set of WAVE files from the Archives New Zealand collection.

   OPF have made the set of preservation masters available to the public here[1]: http://resources.openpreservation.org/dp-stage-boss-one/wavs/. The purpose of this test was to understand the performance of our identification tools vs. checksum generation tools on a group of objects we have found may run counter to the hypothesis that format identification would not take longer than checksum generation.

The following processes were run in order, eleven times, with the first result from each removed to counter any disk caching effects we might find[2]. (Ten runs are acknowledged and presented in the benchmark results):

- SHA1DEEP (64-bit)
- MD5DEEP (64-bit)
- DROID *with* container signatures, maximum bytes to scan: NO LIMIT
- DROID *without* container signatures, maximum bytes to scan: NO LIMIT
- DROID *with* container signatures, maximum bytes to scan: 10 MB
- DROID *without* container signatures, maximum bytes to scan: 10MB
- DROID *with* container signatures, maximum bytes to scan: 65535 bytes
- DROID *without* container signatures, maximum bytes to scan: 65535 bytes
- Siegfried *with* container signatures, maximum BOF scan: NO LIMIT
- Siegfried *without* container signatures, maximum BOF scan: NO LIMIT
- Siegfried *with* container signatures, maximum BOF scan: 10MB
- Siegfried *without* container signatures, maximum BOF scan: 10MB
- Siegfried *with* container signatures, maximum BOF scan: 65535 bytes
- Siegfried *without* container signatures, maximum BOF scan: 65535 bytes

[1] For certain files the public can only access delivery copies via our web interface and must otherwise request preservation master copies.
[2] While I suspect only the first of all the tests needed to be removed from the results, I wanted to be as consistent as possible across each process. More established test-frameworks may have other measures to manage, or even utilize, these results.

The ‘maximum bytes to scan’ setting available in DROID (MBS) will impact wildcard (*) signatures. Given a signature of that type for a file format, if a signature is expected to match a file then DROID will scan the file until a match is found, that is, DROID has the potential to scan the entirety of the file without finding a match. Setting a limit to how far it scans is a method for optimizing DROID’s performance. Its effect on precision will be discussed briefly in the result observations.

Siegfried has a setting to scan a maximum amount from the beginning of the file (BOF), and it is expected the behaviour will be similar. The two tools are not considered entirely equivalent. Identification results are presented below.

A setting of 65535 bytes (0.07 megabytes) for the maximum bytes setting is the out-of-the-box setting in DROID; it is not understood to have a historical meaning, my choice of 10MB is equally arbitrary, and I selected it as a reasonable sized value to capture reasonably sized office productivity type file formats, but was still well within the size of some audio visual formats so we may see some performance gain with hopefully better precision than the DROID default.

Ten results are presented per process and from those the mean time taken plus standard deviation in seconds and minutes.

Machine Configuration

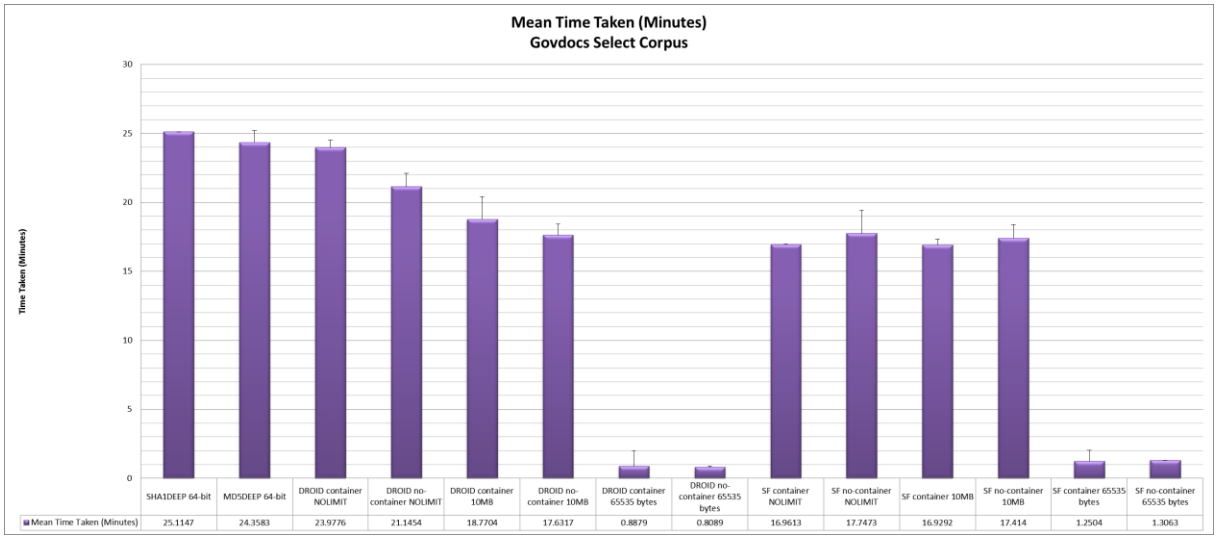
- Intel Core™ i5-3470 CPU @ 3.20GHz
- Western Digital Blue 500GB - 7200 RPM SATA 6 Gb/s 16MB Cache 3.5 Inch - WD5000AAKX
- 16MB Memory
- Java 1.8.0_91-b14

The Results

Govdocs Selected

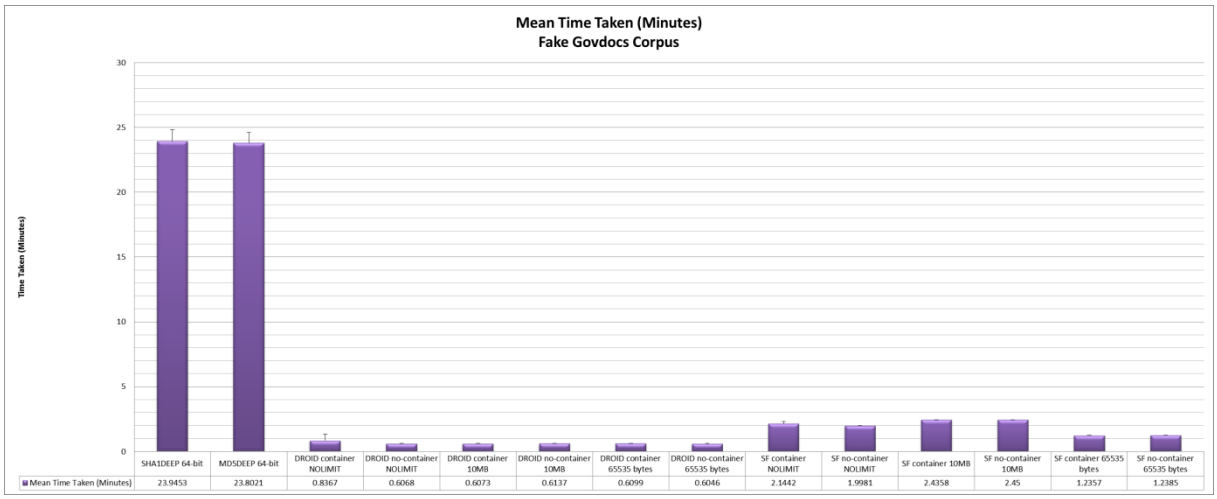
Govdocs Selected is 26,124 files, totalling 31.4GB. At the time of writing[3], DROID can identify approximately 72% of the files inside it. There are approximately 158 file formats identified by DROID.

[3] https://htmlpreview.github.io/?https://github.com/exponential-decay/digital-preservation-stage-boss-one/blob/master/identification-results/droid/droid-analysis-htm/droid-container-NOLIMIT.htm - DROID Analysis HTM, accessed 22 August 2016



The results show that DROID does not take longer to identify the files than the checksum tools take to generate their values. On average, with both DROID and Siegfried, running without maximum byte scan settings, Siegfried runs seven minutes quicker than DROID. DROID runs 2.8 minutes quicker without container signatures. DROID completes its run in less than a minute running at an offset of 65535 bytes (0.07MB). With a ten megabyte MBS DROID is sped up, whereas the difference in Siegfried identification speed is small, (0.03s minutes to 0.3 minutes).

Fake Govdocs

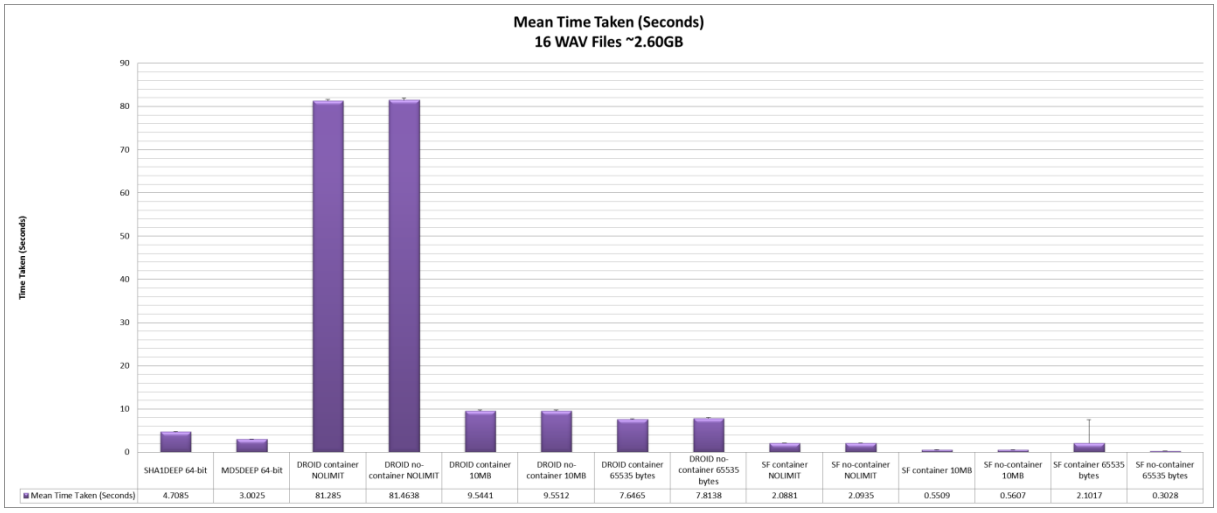


The Fake Govdocs corpus is a control to understand DROID's behaviour when thrown completely random material. It is clear from these graphs that neither Siegfried or DROID waste time with unknown files – that is – they identify unknowns quickly. This is useful behaviour to observe.

With the simulant corpus, Siegfried (a newer tool in comparison to DROID) demonstrates that there may be some performance gains to be found in identifying unknowns quicker.

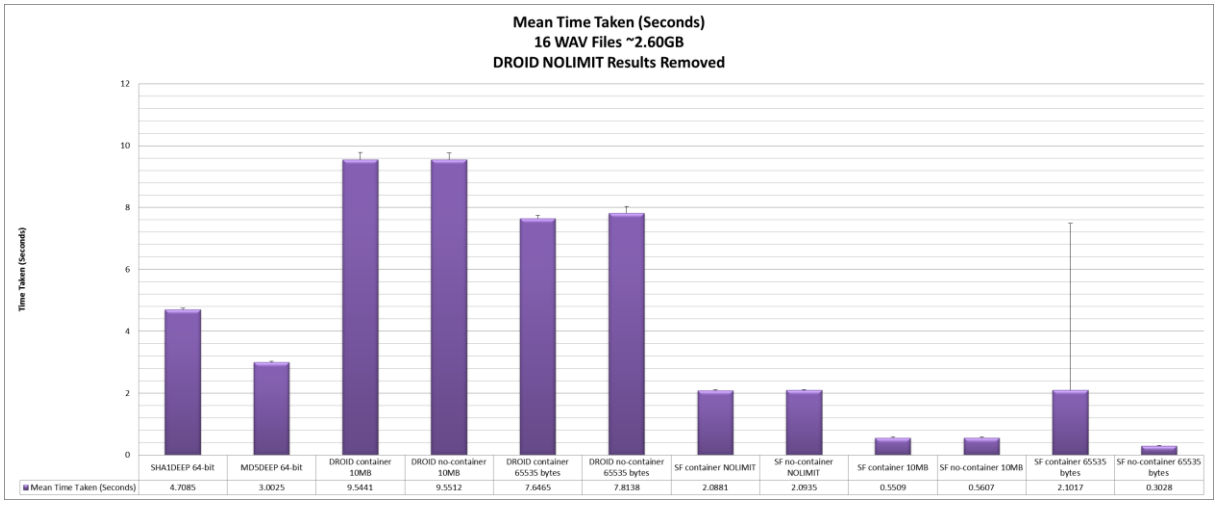
This test demonstrates that only a small amount of time is spent with setup of the Java Virtual Machine and DROID profile setup.

WAVE File Corpus



When we look at the DROID container/no-container NOLIMT results we can see the performance behaviour that inspired these experiments in the first place.

DROID is 16 times slower than the checksum generation process without a maximum byte scan setting. These data points skew the result data so to see DROID's performance better in the other configurations we have to remove them:



With the DROID NOLIMIT results removed it is easier to see 10MB and 65535 byte scans next to the checksum generation timings. DROID still takes longer in both modes with a small standard deviation (66.42 seconds and 2.42 seconds running without container signatures.)

A Siegfried glitch can be seen (Siegfried container 65535 bytes); one result out of the ten takes 18 seconds, while the others nine results show that Siegfried takes within 0.5 seconds. It is expected another process on the PC interfered during this particular scan. I have not corrected for this or felt it necessary to rerun the scan to verify the behaviour of Siegfried when all other things are equal.

Unlike with DROID, the other Siegfried results come in quicker than checksum generation.

Points to note

- Java Virtual Machine (JVM) startup time, and profile generation are included in DROID's performance metrics (DROID's no profile mode would be ideal for these tests but the code available in the repository represents only a simple framework.
- Full benchmark results can be seen in the repository with CSV sheets and an aggregating Excel sheet available: https://github.com/exponential-decay/digital-preservation-stage-boss-one/tree/master/benchmark-results

Impact on File Format Identification

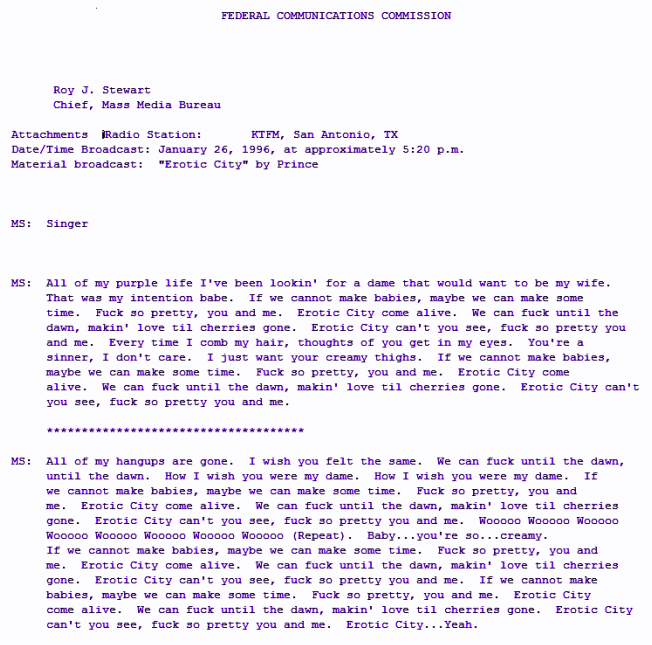
The purpose of this blog was to understand DROID's overall speed in various modes vs. the speed of checksum generation. The tests do allow us to do a little analysis of the impact of a maximum byte scan setting on format identification. The Excel spreadsheet in the repository highlights the differences between settings. I take the unlimited (NOLIMIT) scans for each tool as a baseline, and compare 10MB and 65535 byte scans to that.

I do the same for DROID and Siegfried independently before comparing DROID and Siegfried unlimited scans against each other, using DROID as the benchmark for Siegfried.

Result Observations

- For each decrement in the number of bytes scanned by the tool we see a greater divergence between the DROID NOLIMIT benchmark and that scan's 'identification' results.
- In the three tests comparing DROID using container signatures, the total number of files identified only differs by one:
    - NOLIMIT: *18717*
    - 10MB: *18717*
    - 65535 bytes: *18716*
- This demonstrates that we do see false positives or partial matches being created when DROID's scan length is reduced.

- An example: OLE2 Compound Document Format objects in NOLIMIT mode vs. 65535 bytes is 50 vs. 80. Knowing a file is OLE2 is not as meaningful as knowing it is, say, Microsoft PowerPoint, Serif PagePlus, or Microsoft Excel.
- Between DROID and Siegfried running at their full potential in NOLIMIT mode there are only eight differences in identification. In some instances, this can be explained by incomplete information in PRONOM, or an issue with DROID, see GitHub issues:
    - https://github.com/digital-preservation/droid/issues/106
    - https://github.com/digital-preservation/droid/issues/100
- Siegfried does operate using additional heuristics to identify results as quickly as possible by making more complete use of PRONOM's priority flags, but this is not discussed in more detail here. A full comparison made by Richard Lehane for these sets is available.
- One significant difference in the Comparator tool dataset is 4,748 README files within Govdocs receiving an identification of x-fmt/111 Plain Text File identification by Siegfried. This covers over half of the files left unidentified by DROID (7407). Richard's scan does include file format extension matches. DROID does not contain a text matching engine.
- 23 files reported as plain text (x-fmt/111) by DROID (extension match) are identified as UNKNOWN in Siegfried. An example of the benefit of a text based matcher being available to it.
    - If we inspect these files we understand that it is a positive that Siegfried does not identify these files as text and may indicate risks to their long term preservation, for example: file 381453.txt is notable for a partial transcript of Prince's Erotic City[4]:



FEDERAL COMMUNICATIONS COMMISSION

Roy J. Stewart
Chief, Mass Media Bureau

Attachments Radio Station: KTFM, San Antonio, TX
Date/Time Broadcast: January 26, 1996, at approximately 5:20 p.m.
Material broadcast: "Erotic City" by Prince

MS: Singer

MS: All of my purple life I've been lookin' for a dame that would want to be my wife. That was my intention babe. If we cannot make babies, maybe we can make some time. Fuck so pretty, you and me. Erotic City come alive. We can fuck until the dawn, makin' love til cherries gone. Erotic City can't you see, fuck so pretty you and me. Every time I comb my hair, thoughts of you get in my eyes. You're a sinner, I don't care. I just want your creamy thighs. If we cannot make babies, maybe we can make some time. Fuck so pretty, you and me. Erotic City come alive. We can fuck until the dawn, makin' love til cherries gone. Erotic City can't you see, fuck so pretty you and me.

****

MS: All of my hangups are gone. I wish you felt the same. We can fuck until the dawn, until the dawn. How I wish you were my dame. How I wish you were my dame. If we cannot make babies, maybe we can make some time. Fuck so pretty, you and me. Erotic City come alive. We can fuck until the dawn, makin' love til cherries gone. Erotic City can't you see, fuck so pretty you and me. Wooooo Wooooo Wooooo Wooooo Woooo Wooooo Wooooo (Repeat). Baby...you're so...creamy. If we cannot make babies, maybe we can make some time. Fuck so pretty, you and me. Erotic City come alive. We can fuck until the dawn, makin' love til cherries gone. Erotic City can't you see, fuck so pretty you and me. If we cannot make babies, maybe we can make some time. Fuck so pretty, you and me. Erotic City come alive. We can fuck until the dawn, makin' love til cherries gone. Erotic City can't you see, fuck so pretty you and me. Erotic City...Yeah.

[4] A notice to the federal communications commission about an alleged violation of 18 U.S.C. 1464 by Radio Station KTFM(FM), San Antonio, TX

- But, while you may be able to see this is primarily text, Siegfried's 'UNKNOWN' identification indicates the existence of non-text characters such as ASCII 0x15, the Negative Acknowledge signal: https://en.wikipedia.org/wiki/Acknowledgement_(data_networks)
- For other examples of '*.txt' extension-based matches, if you look deeper into the set, you can examples such as out of ASCII band characters 0x0C (647227.txt) and null (0x00) characters (105297.txt)
- There is scope to use Govdocs for a wider study into content analysis of open corpora. This might include following the trail of identification results such as this to improve identification tools and also to understand the impact of small anomalies like this on preservation.

WAVE Identification Results

WAVE identification results can be found here: https://github.com/exponential-decay/digital-preservation-stage-boss-one/tree/master/identification-results/wav-identification-results.

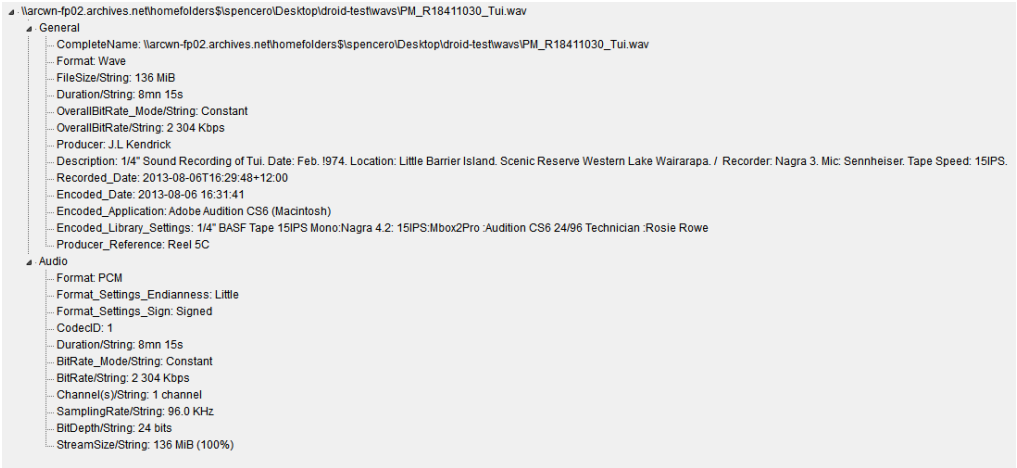
Siegfried (All modes) and DROID with 65535 bytes MBS and 10MB MBS give the result PRONOM fmt/142 (Waveform Audio (WAVEFORMATEX)). DROID NOLIMIT gives multiple identification results, fmt/142, and fmt/704 (Broadcast WAVE 1 PCM Encoding).

I am keen to seek clarification from experts in the audio community, my belief at present, is that the format should be resolved as fmt/704. Broadcast WAVE (BWF) with PCM audio encoding. A 'bext' chunk existing in the file as per the specification[5]:

```
142731392  68 29 62 65 78 74 B8 02 00 00 31 2F 34 22 20 53  h)bext,...1/4" S
142731408  6F 75 6E 64 20 52 65 63 6F 72 64 69 6E 67 20 6F  ound Recording o
142731424  66 20 54 75 69 2E 20 44 61 74 65 3A 20 46 65 62  f Tui. Date: Feb
142731440  2E 20 21 39 37 34 2E 20 4C 6F 63 61 74 69 6F 6E  . !974. Location
142731456  3A 20 4C 69 74 74 6C 65 20 42 61 72 72 69 65 72  : Little Barrier
142731472  20 49 73 6C 61 6E 64 2E 20 53 63 65 6E 69 63 20   Island. Scenic
142731488  52 65 73 65 72 76 65 20 57 65 73 74 65 72 6E 20  Reserve Western
142731504  4C 61 6B 65 20 57 61 69 72 61 72 61 70 61 2E 0D  Lake Wairarapa..
142731520  20 52 65 63 6F 72 64 65 72 3A 20 4E 61 67 72 61   Recorder: Nagra
142731536  20 33 2E 20 4D 69 63 3A 20 53 65 6E 6E 68 65 69   3. Mic: Sennhei
142731552  73 65 72 2E 20 54 61 70 65 20 53 70 65 65 64 3A  ser. Tape Speed:
142731568  20 31 35 49 50 53 2E 00 00 00 00 00 00 00 00 00   15IPS..........
142731584  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
142731600  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
142731616  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
142731632  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
142731648  00 00 00 00 00 00 00 00 00 00 4A 2E 4C 20 4B 65  ..........J.L Ke
142731664  6E 64 72 69 63 6B 00 00 00 00 00 00 00 00 00 00  ndrick..........
142731680  00 00 00 00 00 00 00 00 00 00 52 65 65 6C 20 35  ..........Reel 5
142731696  43 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  C...............
142731712  00 00 00 00 00 00 00 00 00 00 32 30 31 33 2D 30  ..........2013-0
142731728  38 2D 30 36 31 36 3A 33 31 3A 34 31 00 00 00 00  8-0616:31:41....
```

[5] https://tech.ebu.ch/docs/tech/tech3285.pdf - Specification of the Broadcast Wave Format (BWF), accessed 22 August 2016

The information I have from MediaInfo is as follows:



- Normally, a format would not receive a double identification in DROID and it is something that can usually be fixed.
- DROID matches against the two signatures and so it is likely that this is down to no priority information being specified between fmt/142, and fmt/704.
- Although more work needs to be done to investigate this and propose a solution for the PRONOM team; if we look at the record for fmt/142 we can see its relationship as a previous version of fmt/143 (Waveform Audio (WAVEFORMATEXTENSIBLE)). And if we look at fmt/141 (Waveform Audio (PCMWAVEFORMAT)). It's a previous version of fmt/142. The versions are all related.
- In each of fmt/141, fmt/143 we can see a described relationship to the BWF family as well. However, fmt/142 is missing these relationship descriptors:

Fmt/141: Lower Priority than BWF

Name	Waveform Audio (PCMWAVEFORMAT)
Version	
Other names	
Identifiers	MIME: audio/x-wav PUID: fmt/141
Family	WAVE
Classification	Audio
Disclosure	
Description	Waveform Audio (PCMWAVEFORMAT) is the standard Waveform Audio file format, developed by Microsoft. It has now been superseded by Waveform Audio (WAVEFORMATEX), an extended version.
Orientation	Text
Byte order	Little-endian (Intel)
Related file formats	Has lower priority than Broadcast WAVE (1 Generic) Has lower priority than Broadcast WAVE (0 Generic) Has lower priority than Broadcast WAVE (2 Generic) Has lower priority than Broadcast WAVE (0 PCM Encoding) Has lower priority than Broadcast WAVE (1 PCM Encoding) Has lower priority than Broadcast WAVE (2 PCM Encoding) Has priority over Waveform Audio Is previous version of Waveform Audio (WAVEFORMATEX) Is subtype of Waveform Audio

Fmt/142: No BWF Relationship

Name	Waveform Audio (WAVEFORMATEX)
Version	
Other names	
Identifiers	MIME: audio/x-wav PUID: fmt/142
Family	WAVE
Classification	Audio
Disclosure	
Description	Waveform Audio (WAVEFORMATEX) is an extended version of the Waveform Audio file format, and was developed by Microsoft. It has superseded the Waveform Audio (PCMWAVEFORMAT).
Orientation	Text
Byte order	Little-endian (Intel)
Related file formats	Has priority over Waveform Audio Is previous version of Waveform Audio (WAVEFORMATEXTENSIBLE) Is subsequent version of Waveform Audio Is subsequent version of Waveform Audio (PCMWAVEFORMAT) Is subtype of Waveform Audio

Fmt/143: Lower Priority than BWF

Name	Waveform Audio (WAVEFORMATEXTENSIBLE)
Version	
Other names	
Identifiers	MIME: audio/x-wav PUID: fmt/143
Family	WAVE
Classification	Audio
Disclosure	
Description	Waveform Audio (WAVEFORMATEXTENSION) is an extended version of the Waveform Audio file format, and was developed by Microsoft. It is related to the Waveform Audio (WAVEFORMATEX) format, but is an even more extended version.
Orientation	Text
Byte order	Little-endian (Intel)
Related file formats	Has lower priority than Broadcast WAVE (1 Generic) Has lower priority than Broadcast WAVE (0 Generic) Has lower priority than Broadcast WAVE (2 Generic) Has lower priority than Broadcast WAVE (0 WAVEFORMATEXTENSIBLE Encoding) Has lower priority than Broadcast WAVE (1 WAVEFORMATEXTENSIBLE Encoding) Has lower priority than Broadcast WAVE (2 WAVEFORMATEXTENSIBLE Encoding) Has priority over Waveform Audio Is subsequent version of Waveform Audio (WAVEFORMATEX) Is subtype of Waveform Audio

- When we look at the DROID signature file, only one priority relationship is ascribed to all three:

```
<FileFormat ID="784" MIMEType="audio/x-wav"
    Name="Waveform Audio (PCMWAVEFORMAT)" PUID="fmt/141">
    <InternalSignatureID>606</InternalSignatureID>
    <Extension>wav</Extension>
    <Extension>wav</Extension>
    <HasPriorityOverFileFormatID>654</HasPriorityOverFileFormatID>
</FileFormat>
<FileFormat ID="785" MIMEType="audio/x-wav"
    Name="Waveform Audio (WAVEFORMATEX)" PUID="fmt/142">
    <InternalSignatureID>607</InternalSignatureID>
    <Extension>wav</Extension>
    <Extension>wave</Extension>
    <HasPriorityOverFileFormatID>654</HasPriorityOverFileFormatID>
</FileFormat>
<FileFormat ID="786" MIMEType="audio/x-wav"
    Name="Waveform Audio (WAVEFORMATEXTENSIBLE)" PUID="fmt/143">
    <InternalSignatureID>608</InternalSignatureID>
    <Extension>wav</Extension>
    <Extension>wave</Extension>
    <HasPriorityOverFileFormatID>654</HasPriorityOverFileFormatID>
</FileFormat>
```

- It is not clear whether the intent in the BWF relationships being established was to allow DROID to prioritize its results where it removes multiple ID results at the end of scanning using this information - if so it would help DROID avoid this double result.
- Taking a different approach, Siegfried uses priorities as a heuristic to return identification results as quickly as possible. This may be why it returns fmt/142 more definitively – it does not continue further because it does not receive an instruction that there is a better result (a result to prioritize later).
- This means that Siegfried results may also be quicker as per the speed results above due to it narrowing down the result set quicker than perhaps it should if PRONOM were providing this information.
- It may therefore demonstrate the importance of priority settings in PRONOM being set accurately when we create new identification specifications.

Overall Observations

- Some signatures have a big effect on the speed of identification, slowing DROID down beyond the speed of checksum generation
- This may be a cause of wildcard matching on large files - 23 WAVE signatures have wildcards in them which means that if DROID matches the first part of the signature it can continue to try and match the file it is scanning against 23 possible outcomes in parallel until it can discount one by matching the next part of the signature, or the end of the file – whichever comes first.
- A list of all non-container signatures using Wildcards at the time of writing (signature file v86) can be found here: https://github.com/exponential-decay/digital-preservation-stage-boss-one/blob/master/wildcard-signature-information/PRONOM-wildcard-signatures-v86.csv
- We have options for optimization:
    - Within our own workflows, collection based configuration of DROID's max byte scan settings.
    - Create custom signature files per workflow (e.g. well-known digitisation workflows vs. heterogeneous collections for preservation), e.g. using droidsfmin for DROID or ROY for Siegfried.
    - Optimizing signatures using Wildcards by looking for maximum offsets within specifications or deriving average maximum offsets from samples of files.
    - Optimizing the code handling DROID's byte-matching algorithm to cycle through files that might match many signatures in-memory better than we see currently.
    - Optimize unknown detection in Siegfried.

- Focusing on identification precision, an optimization could be to find a better max byte scan setting for DROID, agreed upon in the community. 65535 bytes equates to 0.07MB – this is roughly the same size as an empty OLE2 based Microsoft Word document (Word 2010). For now, this may need to be balanced with the requirements of speed optimization.
- As the speed of DROID's identification is linked to its precision, perhaps, ***no*** maximum byte offset should be set for digital preservation.

Some of the results presented here we need to look at internally, and because we use the Rosetta Digital Preservation system from Ex-Libris we have results we will need to talk about with the Rosetta Community. Currently Rosetta adopts the DROID default settings, though this can be re-configured.

For the larger community I would like to see if we can figure out a way to collectively work through DROID's wildcard signatures to reduce their number as much as feasibly possible. This should at least bring down the average time taken by DROID across heterogeneous content – and we can run these experiments again to find out if it works.

Fixes already in the works…

As you saw above, the results here have translated to the logging of issues in the DROID GitHub repository. The same was done for Siegfried, and Richard Lehane has already rolled up the bug fixes proper in Siegfried 6.1.3: https://github.com/richardlehane/siegfried/blob/master/CHANGELOG.md

Benchmarking for the future…

Chatting to Richard, while I will run these tests another time, it is hoped any other community conversation around this report can be directed toward a more sophisticated, centralised test-harness that can be wrapped around these tools to compare performance in identification speed and results (both positive identification results, and precision) – one that can continue to be run as tools are updated.

Such a harness can help us to see the progress of our work, and provide motivation to see our work progress – for example, being able to identify the 30% of unknowns in the Govdocs Corpus would increase PRONOM coverage for a number of users and we can do this without code changes to either Siegfried or DROID[6]. We can increase speed by reducing wildcards using these non-code based techniques as well.

[6] https://digital-archiving.blogspot.co.nz/2016/08/my-first-file-format-signature.html - My First File Format Signature by Jenny Mitcham, accessed 22 August 2016

Epilogue: Just one more thing…

The title: ***Digital Preservation Stage Boss One***.

I always keep in mind that there are a lot more challenges we are attempting to solve in digital preservation, and potentially many more workflows that need optimizing within the digital preservation life cycle. I simply identified checksum generation and format identification as one of the first 'stage bosses' we need to conquer on the way to better digital preservation systems.

I welcome other experiments like these using public corpora for other processes we might work with along the way to the final confrontation - everything being preserved ☺.

Image via IMDB: http://www.imdb.com/title/tt0316697/mediaviewer/rm1855396608 ©Sega