Skeleton Test Suite: Testing Results
Software Version: skeleton-suite-generator-v0.1-BETA by Ross Spencer

The skeleton test suite provides a mechanism of creating file format 'shells', or, *skeleton* files that test the matching algorithm of the DROID format identification tool and test the integrity and discreteness of DROID compatible format signatures, that is, ensuring a one-to-one (1:1) relationship between a signature and the 'file-format' it matches.

A tool has been created that will take an XML export of the PRONOM database and convert the signatures from the export into these skeleton files. The tool currently navigates the XML structure to find the internal signature section of the document:

```
<InternalSignature>
  <SignatureID>209</SignatureID>
  <SignatureName>Java Compiled Object Code</SignatureName>
  <SignatureNote>Header</SignatureNote>
  <ByteSequence>
    <ByteSequenceID>284</ByteSequenceID>
    <PositionType>Absolute from BOF</PositionType>
    <Offset>0</Offset>
    <MaxOffset>
    </MaxOffset>
    <IndirectOffsetLocation>
    </IndirectOffsetLocation>
    <IndirectOffsetLength>
    </IndirectOffsetLength>
    <Endianness>Little-endian</Endianness>
    <ByteSequenceValue>CAFEBABE</ByteSequenceValue>
  </ByteSequence>
</InternalSignature>
```

This section describes everything we need to know about what DROID is looking for when, in this case, it scans a Java Class (Java Compiled Object Code) file. The key fields we look at here are 'PositionType' and 'ByteSequenceValue'.

Position Type: `Absolute from BOF`
Byte Sequence Value: `CAFEBABE`

This tells DROID to call any binary file with a beginning of file sequence: `0xCAFEBABE`: PUID x-fmt/415 or Java Class File.

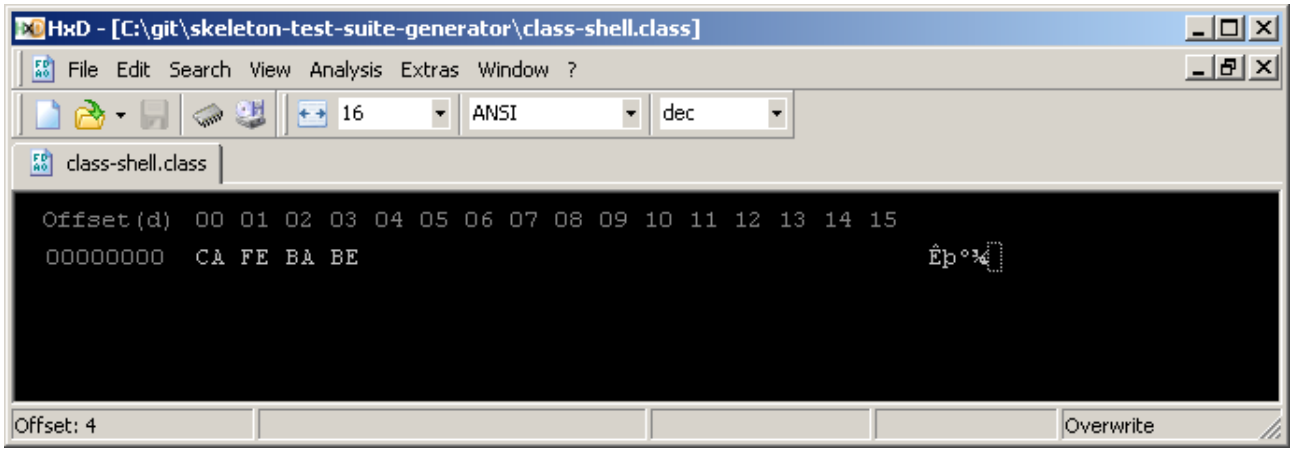

Figure 1: Hex representation of a Java Class file

The skeleton suite generator automatically creates files and outputs binary data into each which will match each byte sequence stored in PRONOM.

As of 15 September 2012, PRONOM is on Signature file v63. In this state the database contains **920 records** which can be exported; 521 of these contain one or more internal signatures, meaning **521 discrete formats** that can be identified using DROID standard signatures. **644 files** are created by the tool; taking into account formats with multiple internal signatures, e.g. MPEG 1/2 Audio Layer 3 (MP3). The tool takes approximately **24.24786 seconds** to create these files using the tool's default settings.

Using the output from the tool we can begin to analyse the content in PRONOM. Preliminary results using the BETA output from the tool are described below. The results are split initially into the problems still to be addressed in the skeleton-test-suite-generator. They then move onto the discussion of further results and issues the skeleton files generated have uncovered in DROID and PRONOM.

NOTE: This report merely begins to report on the complete output and potential benefits of this work, further research centring around the skeleton test suite generator will be released as the tool is better understood and its output refined in future.

Results

The results discussed here are based on the following version of the tool and suite:

- skeleton-test-suite-generator v01-BETA[1]
- skeleton-suite-v0.1-BETA[2]
- DROID 6.1 was used alongside Signature File v63

1 https://github.com/exponential-decay/skeleton-test-suite-generator/tree/v0.1-BETA
2 https://github.com/downloads/exponential-decay/skeleton-test-suite-generator/skeleton-suite-v0.1-BETA.zip

The tool is in its prototype stages. There is some work that needs to be done on it to enable it to handle the byte sequences in PRONOM with 100% accuracy. Basic sequences and well formed and consistent data output from PRONOM is handled correctly. Currently the tool outputs error messages where it may encounter issues. These messages are split into INFO messages and WARNING messages. INFO messages describe where I believe the output should still be correct despite working around issues in the code. WARNING messages describe where skeleton files output should be treated with caution when looking at test results - the hex sequence in these skeleton files might want to be checked manually. The output from the skeleton suite generator where it encounters these messages is described below.

In describing this output we discuss sequence positions using the following acronyms:

- BOF: Beginning of file
- EOF: End of file
- VAR: Variable

BOF with EOF Written

INFO: (fmt/54):	`Attempting to write BOF with EOF written.`
INFO: (fmt/134):	
INFO: (fmt/134):	
INFO: (fmt/134):	
INFO: (fmt/134):	

The generator code has been written in such a way to handle this ordering of sequences in PRONOM. To handle this situation we open up a temporary bytestream in the code, store the existing contents of the skeleton file there, overwrite the skeleton file with the new BOF sequence and then re-output the contents of the bytestream. This has the effect of ordering the two sequences correctly in the skeleton file; the EOF appearing after the BOF. Currently, this technique has only worked for fmt/54 and a handful of the fmt/134 sequences and so should still be considered an outstanding issue in the skeleton suite generator. The code will need to be re-visited to understand why a handful of fmt/134 files generated are not identified by DROID.

The warning in this case represents a scenario that The National Archives may want to look at to help improve the well-formedness of internal signature sequences in PRONOM. A re-ordering of the sequences within the PRONOM database will help to suppress these warnings and mean one less processing stage in outputting skeleton files in future, however, this is very much a nice to have rather than a pressing issue with the PRONOM signature base. It is something to consider if the concept of the skeleton test corpus gains traction.

VAR with VAR written

INFO: (fmt/39):	Attempting to write VAR with VAR written.
INFO: (fmt/40):	
INFO: (fmt/125):	
INFO: (x-fmt/88):	
INFO: (x-fmt/430):	

The formats exhibiting this issue in the generator output are all OLE2 based formats. Looking at the byte steam output by the tool, it seems that the assumption this output would work as expected was incorrect. The tool overwrites the first of the VAR sequences output with the other. In hindsight and looking at the code, we write the VAR sequence from the position immediately after the beginning of file (BOF) sequence was written:

```
self.nt_file.seek(self.boflen)
```

We can work around this trivially by creating a pointer to represent where the first VAR sequence was written. This will initially equal the length of the BOF sequence and after the first var sequence is written it will equal that, and so forth:

```
if self.var_written == False:
    self.var_pos = self.boflen
    self.var_written = True
self.nt_file.seek(self.var_pos)
```

The error output should also be set to WARNING to better reflect the potential pitfalls of writing multiple VAR sequences. We should also attempt to understand whether handling the output of VAR sequences like this represent the essence of the file format we are attempting to simulate, or whether some VAR sequences should actually mandate some sort of ordering depending on the file format – to help tie it down further.

VAR with EOF written

INFO: (fmt/161):	Attempting to write VAR with EOF written.

Again the code is written to buffer existing content in the file, in this case anything after a BOF pointer, if it exists, up to the end of the EOF sequence. This allows us to output a VAR sequence between the two before reading the buffer back into the skeleton file. The National Archives may wish to help in suppressing this warning by re-ordering signature elements exhibiting this behaviour in PRONOM, however, only SIARD (fmt/161) exhibits this currently and is generated correctly by the tool to be identified by DROID.

BOF with BOF written (offset > BOF pointer)

WARNING: (fmt/363):	`Attempting to write BOF with BOF written. Attempting to correct: offset > current BOF file pointer...`
WARNING: (fmt/363):	
WARNING: (x-fmt/387):	
WARNING: (x-fmt/387):	
WARNING: (x-fmt/388):	
WARNING: (x-fmt/399):	
WARNING: (x-fmt/399):	

This message represents where a second BOF sequence is encountered. The offset of the second is greater than the length of the first and so we are able to calculate where to write the sequence and output it normally. In all cases above the sequences are output correctly and the skeleton files thusly identified by DROID. PRONOM states the reason for the multiple BOF sequences existing in some of these cases is because of a bug in DROID relating to the longest fragment of a signature needing to exist closest to the anchor (i.e. BOF or EOF) of that signature, for example this text from the EXIF 2.0 signature, x-fmt/399:

Description: TIFF header, EXIF IFD near BOF (EXIF ver=2.0) *[NB: When the DROID bug associated with longest fragment not nearest anchor is fixed, the 2 BOF sequences can be merged.]*
Byte sequences
Position type: `Absolute from BOF`
Offset: `0`
Value: `4D4D002A`
Position type: `Absolute from BOF`
Offset: `10`
Maximum Offset: `65535`
Value: `900000070000000430323030`

While it seems the PRONOM team have a suitable workaround for this issue it would be useful to see them fix it as soon as possible to increase the confidence with which signature developers can work with the syntax given to them by The National Archives.

A benefit of the skeleton suite methodology is shown here. Unit tests can be used during the development of a solution for this issue and the developers of DROID need only work with a byte sequence which demonstrates the problem, e.g.:

Position type: `Absolute from BOF`
Offset: `0`
Maximum Offset: `65535`

Value: `4D4D002A{10}900000070000000430323030`

The skeleton file will not be identified initially but it can be used for debugging the code. Using test-driven development the team need only to see the file/s eventually be identified by DROID and they can be confident the issue has been fixed.

BOF with BOF written (offset == zero)

WARNING: (fmt/189):	`Attempting to write BOF with BOF written. Attempting to correct: offset == zero so writing after...`
WARNING: (fmt/358):	
WARNING: (fmt/358):	
WARNING: (fmt/358):	
WARNING: (x-fmt/388):	

Given the scenario where we try to write two BOF sequences to a file, both of which have an offset of zero, there is little we can do in code to 'guess' where to output the second byte steam. As such, the code outputs it immediately after the BOF sequence already written.

In the case of fmt/189, PRONOM describes two BOF sequences. In order, the first is described as a BOF with a 30 byte offset. The second sequence is a BOF with an offset of zero. Without handling this signature specifically or creating a more sophisticated pre-processing mechanism in our code there is little we can do currently to organise this output correctly in the skeleton file. As such, the fmt/189 skeleton file does not get written correctly. The second sequence overwrites the first. The National Archives may wish to consider joining these two BOF sequences together as semantically the idea of two BOF sequences does not seem correct, however it should be observed that it seems that the ability to do this depends on the ability for the PRONOM team to be able to tie down the wild cards found in the second sequence with the zero offset:

```
50 4B 03 04 * 50 4B 01 02 * 50 4B 05 06
```

Fmt/358 (Internet Data Query File) represents a complicated signature with four BOF sequences which can appear in any order within the file within the first zero to 3424 bytes. I don't believe this signature can be improved or re-written. We should simply continue to output the warning and monitor that this file and similar files are output correctly in future.

X-fmt/388 (EXIF 2.1) records four different internal signatures, one of which has two BOF sequences each with an offset of zero. Again, PRONOM states the reason for the multiple BOF sequences existing is because of a bug in DROID relating to the longest fragment of a signature needing to exist closest to the anchor (i.e. BOF or EOF) of that signature:

Description: TIFF header, EXIF IFD near BOF (EXIF ver=2.1). [NB: When the DROID bug associated

with longest fragment not nearest anchor is fixed, the 2 BOF sequences can be merged.]

Byte sequences

Position type: `Absolute from BOF`

Offset: `0`

Value: `4D4D002A`

Position type: `Absolute from BOF`

Offset: `0`

Value: `{10-65535}900000070000000430323130`

Further skeleton suite testing results

Beyond the deficiencies of the generator tool described above, we were able to output **100%** of internal signatures in PRONOM as skeleton files **(644 of 644)** - as will be discussed in the results below, **92%** describe uniquely matched files that are accurate to the signatures described in PRONOM **(592 of 644)**. Using the complete set of 644 files we can observe the behaviour of DROID's signature matching capabilities to find areas which need to be improved in the two tools provided by The National Archives.

Files that *should* be identified by DROID

The most concerning result found using the skeleton test suite were files generated that further research showed should be matched to a PUID in DROID.

The skeleton file for fmt/126 output by the generator (Powerpoint 97-2002) is identified as fmt/111 (OLE2). Having reviewed the signature in PRONOM and created an equivalent skeleton file manually, it is unclear why DROID does not identify the file correctly. It is believed by this report that the signature should match the file and therefore should be reviewed by the team at The National Archives as soon as they are capable of doing so.

Similarly, x-fmt/412 is a straightforward internal signature that looks to output correctly from the skeleton-suite-generator. The deeper analysis of the output and signature and manual generation of an equivalent skeleton file show that this file should be identified by DROID but it isn't.

With both of these results, a sanity check by The National Archives would prove useful. A further clue that DROID 6.1 is not handling the identification of these files correctly is that they are correctly matched in DROID 4.0 which was the last iteration of DROID before larger changes were made to the identification engine. The National Archives should seek to understand the reasons for this issue as soon as possible.

A further benefit of the skeleton test suite is demonstrated by these two files - the need for regression testing is important, however, it is recognised that the regression testing of format identification this is not trivial

without access to a complete test suite of file formats. The skeleton suite replaces this need in the short term. Using the skeleton corpus of files, unit tests existing between DROID 4.0 and 6.1 would have revealed the breaking of the identification of these two formats, which may in turn reveal a deeper issue within the matching algorithm in DROID. Given the benefits of identifying issues such as this, the incorporation of the test-suite should be considered in future versions of the DROID source code.

The skeleton file for fmt/435 represents an issue where unit tests would not have improved the situation as this file is not identified in either DROID 4.0 or 6.1. Having analysed this file and the signature it is unclear why it won't match. A sanity check by The National Archives would prove useful and it is hoped that the error is an issue with the generator rather than DROID itself, although it is difficult to see where the problem lies in the file at present.

Mismatched file name and identifications:

Reviewing the identification results returned by DROID on the skeleton test suite, we find that fmt/60 and fmt/61 are identified as fmt/59 and fmt/62 respectively. Looking at PRONOM fmt/60 and fmt/59 are assigned the same signature. It isn't clear what determines which signature is returned but it is likely to be the assigned relationships between the two. This is the same scenario with fmt/61 and fmt/62. The PRONOM team should seek to understand whether these are in fact distinct formats, or whether something can be done to distinguish the two by modifying these byte sequences rather than attaching the same internal signatures to the two sets of records.

DROID 6.1 Nuances

The following open document based formats are all generated by the skeleton-suite-generator.

- fmt/135 – Open Document Text 1.0
- fmt/290 – Open Document Text 1.1
- fmt/291 – Open Document Text 1.2
- fmt/140 – Open Document Database 1.0
- fmt/138 – Open Document Presentation 1.0
- fmt/292 – Open Document Presentation 1.1
- fmt/293 – Open Document Presentation 1.2
- fmt/137 – Open Document Spreadsheet 1.0
- fmt/294 – Open Document Spreadsheet 1.1
- fmt/295 – Open Document Spreadsheet 1.2

The byte sequences for each of these formats is straightforward for the tool to handle. Manual checking shows the generation of a matching skeleton file to be accurate. This is also backed up by the identification

of these skeleton files as the correct PUIDs within DROID 4.0.

In DROID 6.1 these formats are either identified by extension only or are given no identification at all. The reason is again unclear and should be understood whether this is expected in DROID or an issue in the identification engine. My hypothesis is that due to the container identification mechanism employed by DROID 6.01 upwards, the identification of these files is escalated to a different identification mechanism. As these are not 'container' files in the strictest sense they remain unidentified by the container mechanism and their identification is not handed back to the standard identification engine employed by DROID.

I feel it would be useful for the community to have the algorithm for this non-match clarified in the DROID documentation. Further, a question which might be raised for future developments of DROID is 'should there be better feedback by the tool?' Given that a standard signature and set of byte sequences exist to provide some sort of rudimentary identification of these formats - should DROID hand the identification of these files back the standard engine and return 'what it can'? Benefits might include providing a *clue* to unknown binary streams in one's possession, maybe even potentially identifying files that might have become corrupted, where open document format container signatures wouldn't match but these standard signatures might. There may even be an argument for this approach to simply handle occasions where the signatures recorded in the container signature file do not capture all potential instances of Open Document based format.

Additional feedback from the tool does seem feasible, for example providing both the identification as returned by the standard identification engine and the identification returned by the container identification if two types of signatures exist.

Issues identified with PRONOM

Version 0.1 of the skeleton suite also helped show a handful of issues with the records in PRONOM, summarised below:

- x-fmt/451: The same signature is given to this record as x-fmt/452 in PRONOM so there will always be a duplicate identification. The version information (n/a) in the PRONOM record may also want to be considered for re-wording.
- x-fmt/452: As above, this record shares the same signature as fmt/451. The version is also documented as 'n/a'.
- x-fmt/34: The same signature is given to this record as x-fmt/35 in PRONOM so there will always be a duplicate identification.
- x-fmt/35: As above, this record shares the same signature as x-fmt/34.
- x-fmt/116: The same signature is given to this record as x-fmt/212 in PRONOM so there will always be a duplicate identification.
- x-fmt/212: As above, this record shares the same signature as x-fmt/116.

- fmt/414: The same signature is given to this record as x-fmt/135 in PRONOM so there will always be a duplicate identification.
- x-fmt/135: As above, this record shares the same signature as fmt/414.
- fmt/91 and fmt/92: SVG, the wild card between the <svg> tags described in the signature means that fmt/91 signatures will also match fmt/92 files.
- fmt/373 and fmt/381: fmt/373 signature is less verbose and so can match fmt/381 conformant files resulting in duplicate identification.
- fmt/11 and fmt/13: fmt/11 describes a subsequence of fmt/13 so will match fmt/13 conformant files.
- fmt/141 and fmt/142: fmt/141 describes a subsequence of fmt/142 which results in duplicate identification.
- fmt/353 and fmt/436: fmt/353 describes a subsequence of fmt/436 so there will always be duplicate identification with these files.
- fmt/131 and fmt/441: fmt/131 describes a subsequence of fmt/441 so there will always be duplicate identification of these files.

Signatures describe a de-facto specification of a file format. Taking these signatures and outputting binary data that matches those signatures allows us to create files which describe the *essence* of the identity of a 'unique' file format. If two file formats are given the same signatures then it highlights a potential mistake in the distinction/granularity made between two formats in PRONOM and in any format registry. In adopting a signature-centric approach to populating PRONOM[3] The National Archives are in a position to avoid such issues moving forward but it does require the correction of existing issues in the short term. While the granularity of the PUIDs which exist for these files *may* be correct and the signatures potentially corrected to reflect this, either by way of setting priorities in PRONOM, or amending byte sequences as appropriate – it is demonstrated with the initial results of this testing that the skeleton test corpus approach provides a mechanism for The National Archives and the community to monitor and correct these *initial* issues and keep a handle on them going into the future, ensuring the correct granularity and discreteness of identification in format identification tools.

While the eyeball checking and testing with limited number of files is useful, the existence of test files, either skeleton test suite files or a fully fledged file format test suite allows The National Archives and the community to potentially automate the process of checking the quality and consistency for format signatures, potentially putting signature development on a par with continuous integration processes used in software development.

Despatches

RTF (Rich Text Format) files generated by the skeleton suite generator are given multiple identifications in DROID. This is a long-standing issue in the PRONOM database and no attempt has been made here to understand the files generated by the tool. With the weakness lying in the signatures themselves it is

3 http://blog.nationalarchives.gov.uk/blog/an-introduction-to-the-pronom-contribution-model-and-the-signature-developer-role/

expected that the RTF signatures will either one day be consolidated into one by the PRONOM team, or identified by text-identification techniques as DROID functionality is added to in future.

Conclusion

The results described in this paper are merely the results of the first iteration of the skeleton-test-suite-generator. Although extensive I do not believe the results to be exhaustive. While I have attempted to outline most issues found with the generator tool and PRONOM/DROID, narrowing down bugs and errors is an iterative process. As we fix issues discussed in this paper new problems may arise or existing issues will become more visible.

The results here will now be taken to improve the skeleton-test-suite-generator output; used to discuss how we move this work forward and will inform the generation of a managed, hosted, skeleton-test-suite online. The official release of a skeleton test suite should incorporate an initial output from the test suite generator to take care of the hundreds of files in PRONOM at present but it should then be augmented by adding *manually* created files. This will be discussed in future research as this topic is already beyond the scope of this results paper.

Hopefully the continual refinement of this work will continue to prove useful to improving the content of PRONOM and the identification capabilities of DROID. It is a hope of this paper that The National Archives will be open to correspondence about the results discussed here and I can present this correspondence and the follow up actions in future work related to the *skeleton test corpus*.