2.1 External signatures

External signatures encompass all format indicators which are external to the object bitstream, such as Macintosh data forks and Windows file extensions.

In some operating systems, such as Microsoft DOS and Windows, an external signature is provided by the file extension. This indicates the format of the object, e.g. Mydoc.doc for a Microsoft Word document, and Mypic.tif for a TIFF image. However, the primary function of the extension is not to identify the format, but rather to indicate to the operating system the default software package which should be used to open the file. As a result, file extensions suffer from three disadvantages as a method of identification:

- Extensions are not necessarily unique to a single format. For example, the .wks extension is used for both Lotus 1-2-3 worksheets and Microsoft Works documents.

- They do not provide sufficient granularity to adequately identify format versions. For example, the .doc extension does not distinguish between a Word 6 document and a Word 2003 document, although these are significantly different formats.

- Extensions can be defined or altered by users. For example, a user of WordPerfect 5.1 might choose to distinguish letters and memos by using .let and .mem extensions respectively, overriding the default extension assigned by the system.

External signatures are best suited to providing a general indication of the file format, and should not be relied upon for definitive identification. PRONOM records known external signatures which are associated with a given format. However, in DROID, these are accorded much less weight than internal signatures in the identification process. External signatures are described in PRONOM as type/value pairs; currently file extensions are the only permitted type.

2.2 Internal signatures

Internal signatures encompass all format indicators which are contained within the object bitstream. By definition, a file format specification imposes a specific structure upon the content of the bitstream, which is consistent between all digital objects in that format. The characteristics of this structure may therefore be used as a signature for identifying the format. Some formats, such as PNG, include a signature specifically designed to allow identification; in other cases, an incidental signature can be derived from elements of the format structure. Longer signature sequences are clearly preferable to shorter ones, since they reduce the possibility of false identification through coincidental sequences appearing in a bitstream. Even a deliberate signature, such as PNG, may be enhanced using additional structural elements, to provide a more secure identification.

In PRONOM, an internal signature is composed of one or more byte sequences, each comprising a continuous sequence of hexadecimal byte values and, optionally, regular expressions. A signature byte sequence is modelled by describing its starting position within a bitstream and its value. The starting position can be one of two basic types:

- Absolute: the byte sequence starts at a fixed position within the bitstream. This position is described as an offset from either the beginning or the end of the bitstream. The byte sequence can therefore be located by moving to the specified offset, counting from either the Beginning of File (BOF) or End of File (EOF) position. If counting from the EOF position, the offset is to the final byte in the sequence.

- Variable: the byte sequence can start at any offset within the bitstream. The byte sequence can be located by examining the entire bitstream.

The value of the byte sequence is defined as a sequence of hexadecimal values, optionally incorporating any of the following regular expressions:

- **??:** wildcard matching any pair of hexadecimal values (i.e. a single byte).

  e.g.: 0x0A FF ?? FE would match 0x0A FF 6C FE or 0x0A FF 11 FE.

- ***:** wildcard matching any number of bytes (0 or more).

  e.g.: 0x0A FF * FE would match 0x0A FF 6C FE or 0x0A FF 6C 11 FE.

- **{n}:** wildcard matching *n* bytes, where *n* is an integer.

  e.g.: 0x1C 20 {2} 4E 12 would match 0x1C 20 FF 15 4E 12.

- **{m-n}:** wildcard matching between *m*-*n* bytes inclusive, where *m* and *n* are integers or '*'.

  e.g.: 0x03 {1-2} 4D would match 0x03 3C 4D or 0x03 3C 88 4D.

  e.g.: 0x03 {2-*} 4D would match 0x03 3C 88 4D or 0x03 3C 88 3F 4D.

- **(a|b):** wildcard matching one from a list of values (e.g. *a* or *b*), where each value is a hexadecimal byte sequence of arbitrary length containing no wildcards.

  e.g.: 0x0E (FF|FE) 17 would match 0x0E FF 17 or 0x0E FE 17.

- **[a:b]:** wildcard matching any sequence of bytes which lies lexicographically between *a* and *b*, inclusive (where both *a* and *b* are byte sequences of the same length, containing no wildcards, and where *a* is less than *b*). The endian-ness of *a* and *b* are the same as the endian-ness of the signature as a whole.

  e.g. 0xFF [09:0B] FF would match 0xFF 09 FF, 0xFF 0A FF or 0xFF 0B FF.

- **[!a]:** wildcard matching any sequence of bytes other than *a* itself (where *a* is a byte sequence containing no wildcards).

  e.g. 0xFF [!09] FF would match 0xFF 0A FF, but not 0xFF 09 FF.

- **[!a:b]:** wildcard matching any sequence of bytes which does not lie lexicographically between *a* and *b*, inclusive (where *a* and *b* are both byte sequences of the same length, containing no wildcards, and where *a* is less than *b*).

    e.g. 0xFF [!01:02] FF would match 0xFF 00 FF and 0xFF 03 FF, but not 0xFF 01 FF or 0xFF 02 FF.

Note: In the examples above, spaces are included between byte values for reasons of clarity, but are omitted in actual byte sequence values. The signature is processed left-to-right if the signature is measured relative to BOF and right-to-left if measured relative to EOF. The endian-ness of the signature is only relevant for sequences inside square brackets.

A byte sequence must contain a fixed subsequence of at least one byte between each occurrence of '*', or between the beginning or end of the sequence and an occurrence of '*'. Thus, sequences of the following form are not permitted:

[BOF] (a|b)*…

…*(a|b) [EOF]

…*(a|b)*…

The syntax of a byte sequence may be expressed in formal BNF notation as follows:

```
<byte_sequence> ::= <fixed_subsequence> | <byte_sequence>
<variable_wildcard> <fixed_subsequence>

<fixed_subsequence> ::= <chunk> | <fixed_subsequence> <chunk>

<chunk> ::= <byte_subsequence> | <wildcard>

<wild_card> ::= '??' | '{' <integer> '}' | '{' <integer> '-'
<integer> '}' | '(' <byte_subsequence> '|' <byte_subsequence>
')' | '[' <byte_subsequence> ':' <byte_subsequence> ']' | '[!'
<byte_subsequence> ':' <byte_subsequence> ']' | '[!'
<byte_subsequence> ']'

<variable_wildcard> ::= '*' | '{' <integer> '-' '*' '}'

<byte_subsequence> ::= <byte> | <byte_subsequence> <byte>

<byte> ::= <hexadecimal_digit> <hexadecimal_digit>

<hexadecimal_digit> ::= <digit> | <letter>

<integer> ::= <digit> | <integer> <digit>

<digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8'
| '9'
```

```
<letter> ::= ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’
```

The order in which byte sequences are recorded within a signature is arbitrary, and does not imply any order or prioritisation within the bitstream. For example, if a signature contains two variable-position sequences, these may appear in any order within a given bitstream. In cases where order is significant (e.g. Sequences 1 and 2 are both variable-position, but Sequence 2 must appear after Sequence 1 in the bitstream), this would be recorded by combining the relevant sequences into a single sequence, separated by the appropriate regular expressions. For the purposes of identification, a Boolean ‘And’ relationship is assumed between every sequence in a signature, i.e. a given object must match every sequence in the signature to match the signature.

2.2.1 Signature specificity

The level of granularity at which internal signatures can be defined may vary. In many cases, it is possible to define internal signatures which are specific to a single version of a given format. However, in other cases a single signature may be common to more than one version of a format (e.g. TIFF), or even to a class of formats (e.g. the OLE2 Compound Document Format). Signatures which are unique to one format record in PRONOM are termed ‘specific’ signatures, whereas those which are common to multiple format records are termed ‘generic’ signatures. No format may have both a generic and a specific internal signature. This distinction is reflected in the identification result: a match with a specific signature is recorded as a ‘positive specific’ identification, and a match with a generic signature is recorded as a ‘positive generic’ identification.

2.2.2 Format priority

It is possible for subtype and supertype relationships to exist between formats. For example, SVG can be considered as a subtype of XML. Since an SVG document is also an XML document, it would match the internal signatures for both formats. However, it is desirable that only the most specific identification be reported. To enable this, the PRONOM data model allows a ‘priority’ relationship to be defined between two formats. Where such a relationship exists, and a given object matches signatures for both formats, the lower priority identification is discarded.

2.2.3 Example byte sequences

The possible scenarios can be illustrated as follows (note that all offsets are calculated starting from 0):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1																					

Byte sequence 1 occurs at an absolute offset from the BOF. It would therefore be described as follows:

System ID: 1
Position Type: Absolute from BOF
Offset: 0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
															2						

Byte sequence 2 occurs at an absolute offset from the EOF. It would therefore be described as follows:

System ID: 2
Position Type: Absolute from EOF
Offset: 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
				3																	

Byte sequence 3 occurs at a variable offset. It would therefore be described as follows:

System ID: 3
Position Type: Variable
Offset: n/a

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
				4a					**4b**												

Byte sequence 4 includes two subsequences. Sequence 4a appears at a variable offset, and 4b appears at a fixed relative offset to 4a. It would therefore be described as follows:

System ID: 4
Position Type: Variable
Offset: n/a
Value: *4a*{2}*4b*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
				5a										**5b**							

Byte sequence 5 includes two subsequences. Sequence 5a appears at an absolute offset, and 5b appears at a variable relative offset to 5a. It would therefore be described as follows:

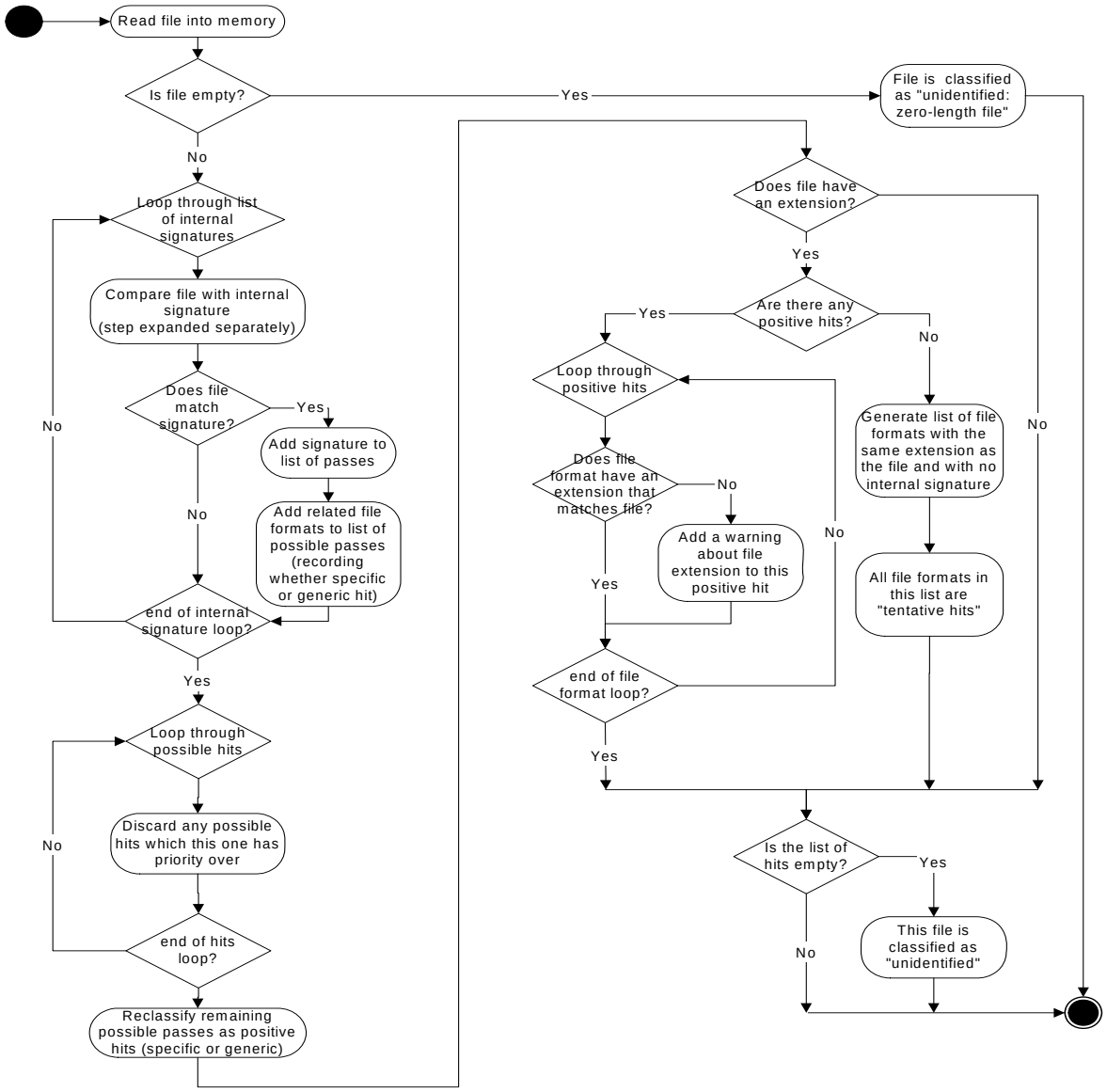
System ID: 5
Position Type: Absolute from BOF
Offset: 4
Value: 5*a*{*}5*b*

3 The DROID format identification process

The DROID software tool uses signature information stored in PRONOM to perform automatic format identification. This section describes the identification algorithm employed by DROID, and the formats of the XML files used by DROID to describe signatures and record the results of the identification process.

3.1 The format identification algorithm

The format identification process employed by DROID is illustrated in the following activity diagram:



If a file is too big to hold in memory in its entirety, the first step is skipped. Instead, the file is read into memory a chunk at a time. To be specific, whenever the algorithm needs to access byte *n* of the file, a check is made to see whether byte *n* is already in memory. If