PNG (Portable Network Graphics) Specification, Version 1.2

3. File Structure

A PNG file consists of a PNG *signature* followed by a series of *chunks*. This chapter defines the signature and the basic properties of chunks. Individual chunk types are discussed in the next chapter.

3.1. PNG file signature

The first eight bytes of a PNG file always contain the following (decimal) values:

```
137 80 78 71 13 10 26 10
```

This signature indicates that the remainder of the file contains a single PNG image, consisting of a series of chunks beginning with an IHDR chunk and ending with an IEND chunk.

See Rationale: PNG file signature.

3.2. Chunk layout

Each chunk consists of four parts:

Length
: A 4-byte unsigned integer giving the number of bytes in the chunk's data field. The length counts **only** the data field, **not** itself, the chunk type code, or the CRC. Zero is a valid length. Although encoders and decoders should treat the length as unsigned, its value must not exceed 2^{31} bytes.

Chunk Type
: A 4-byte chunk type code. For convenience in description and in examining PNG files, type codes are restricted to consist of uppercase and lowercase ASCII letters (A-Z and a-z, or 65-90 and 97-122 decimal). However, encoders and decoders must treat the codes as fixed binary values, not character strings. For example, it would not be correct to represent the type code IDAT by the EBCDIC equivalents of those letters. Additional naming conventions for chunk types are discussed in the next section.

Chunk Data
: The data bytes appropriate to the chunk type, if any. This field can be of zero length.

CRC
: A 4-byte CRC (Cyclic Redundancy Check) calculated on the preceding bytes in the chunk, including the chunk type code and chunk data fields, but **not** including the length field. The CRC is always present, even for chunks containing no data. See CRC algorithm.

The chunk data length can be any number of bytes up to the maximum; therefore, implementors cannot assume that chunks are aligned on any boundaries larger than bytes.

Chunks can appear in any order, subject to the restrictions placed on each chunk type. (One notable restriction is that IHDR must appear first and IEND must appear last; thus the IEND chunk serves as an end-of-file marker.) Multiple chunks of the same type can appear, but only if specifically permitted for that type.

See Rationale: Chunk layout.

3.3. Chunk naming conventions

Chunk type codes are assigned so that a decoder can determine some properties of a chunk even when it does not recognize the type code. These rules are intended to allow safe, flexible extension of the PNG format, by allowing a decoder to decide what to do when it encounters an unknown chunk. The naming rules are not normally of interest when the decoder does recognize the chunk's type.

Four bits of the type code, namely bit 5 (value 32) of each byte, are used to convey chunk properties. This choice means that a human can read off the assigned properties according to whether each letter of the type code is uppercase (bit 5 is 0) or lowercase (bit 5 is 1). However, decoders should test the properties of an unknown chunk by numerically testing the specified bits; testing whether a character is uppercase or lowercase is inefficient, and even incorrect if a locale-specific case definition is used.

It is worth noting that the property bits are an inherent part of the chunk name, and hence are fixed for any chunk type. Thus, BLOB and bLOb would be unrelated chunk type codes, not the same chunk with different properties. Decoders must recognize type codes by a simple four-byte literal comparison; it is incorrect to perform case conversion on type codes.

The semantics of the property bits are:

Ancillary bit: bit 5 of first byte
: 0 (uppercase) = critical, 1 (lowercase) = ancillary.

  Chunks that are not strictly necessary in order to meaningfully display the contents of the file are known as "ancillary" chunks. A decoder encountering an unknown chunk in which the ancillary bit is 1 can safely ignore the chunk and proceed to display the image. The time chunk (tIME) is an example of an ancillary chunk.

  Chunks that are necessary for successful display of the file's contents are called "critical" chunks. A decoder encountering an unknown chunk in which the ancillary bit is 0 must indicate to the user that the image contains information it cannot safely interpret. The image header chunk (IHDR) is an example of a critical chunk.

Private bit: bit 5 of second byte
: 0 (uppercase) = public, 1 (lowercase) = private.

  A public chunk is one that is part of the PNG specification or is registered in the list of PNG special-purpose public chunk types. Applications can also define private (unregistered) chunks for their own purposes. The names of private chunks must have a lowercase second letter, while public chunks will always be assigned names with uppercase second letters. Note that decoders do not need to test the private-chunk property bit, since it has no functional significance; it is simply an administrative convenience to ensure that public and private chunk names will not conflict. See Additional chunk types, and Recommendations for Encoders: Use of private chunks.

Reserved bit: bit 5 of third byte
: Must be 0 (uppercase) in files conforming to this version of PNG.

  The significance of the case of the third letter of the chunk name is reserved for possible future expansion. At the present time all chunk names must have uppercase third letters. (Decoders should not complain

about a lowercase third letter, however, as some future version of the PNG specification could define a meaning for this bit. It is sufficient to treat a chunk with a lowercase third letter in the same way as any other unknown chunk type.)

Safe-to-copy bit: bit 5 of fourth byte

0 (uppercase) = unsafe to copy, 1 (lowercase) = safe to copy.

This property bit is not of interest to pure decoders, but it is needed by PNG editors (programs that modify PNG files). This bit defines the proper handling of unrecognized chunks in a file that is being modified.

If a chunk's safe-to-copy bit is 1, the chunk may be copied to a modified PNG file whether or not the software recognizes the chunk type, and regardless of the extent of the file modifications.

If a chunk's safe-to-copy bit is 0, it indicates that the chunk depends on the image data. If the program has made *any* changes to *critical* chunks, including addition, modification, deletion, or reordering of critical chunks, then unrecognized unsafe chunks must **not** be copied to the output PNG file. (Of course, if the program **does** recognize the chunk, it can choose to output an appropriately modified version.)

A PNG editor is always allowed to copy all unrecognized chunks if it has only added, deleted, modified, or reordered *ancillary* chunks. This implies that it is not permissible for ancillary chunks to depend on other ancillary chunks.

PNG editors that do not recognize a *critical* chunk must report an error and refuse to process that PNG file at all. The safe/unsafe mechanism is intended for use with ancillary chunks. The safe-to-copy bit will always be 0 for critical chunks.

Rules for PNG editors are discussed further in Chunk Ordering Rules.

For example, the hypothetical chunk type name bLOb has the property bits:

```
bLOb  <-- 32 bit chunk type code represented in text form
||||
|||+- Safe-to-copy bit is 1 (lowercase letter; bit 5 is 1)
||+-- Reserved bit is 0     (uppercase letter; bit 5 is 0)
|+--- Private bit is 0      (uppercase letter; bit 5 is 0)
+---- Ancillary bit is 1    (lowercase letter; bit 5 is 1)
```

Therefore, this name represents an ancillary, public, safe-to-copy chunk.

See Rationale: Chunk naming conventions.

3.4. CRC algorithm

Chunk CRCs are calculated using standard CRC methods with pre and post conditioning, as defined by ISO 3309 [ISO-3309] or ITU-T V.42 [ITU-T-V42]. The CRC polynomial employed is

```
x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1
```

The 32-bit CRC register is initialized to all 1's, and then the data from each byte is processed from the least significant bit (1) to the most significant bit (128). After all the data bytes are processed, the CRC register is inverted (its ones complement is taken). This value is transmitted (stored in the file) MSB first. For the purpose of separating into bytes and ordering, the least significant bit of the 32-bit CRC is defined to be the coefficient of the x^{31} term.

Practical calculation of the CRC always employs a precalculated table to greatly accelerate the computation. See Sample CRC Code.