Problem

The meaning of the safe-to-copy bit is currently contested.
As all new chunks must include the safe-to-copy bit (either set or not), the definition must be made clear.

The PNG spec has rules around the ordering of chunks given their safe-to-copy bit. Those rules might need to be updated in a future edition. However, Third Edition introduces new chunks which must have their safe-to-copy bit either set or not using the existing definition.

The meaning of “PNG editor” is highly correlated with the safe-to-copy bit. So its meaning is also currently contested and needs to be resolved.

Background

Each chunk has a 4-letter name. Those letters being upper-case or lower-case carries meaning. The 4th letter’s casing indicates if the chunk is safe-to-copy when the PNG editor does not know the chunk.

If a PNG editor understands the given chunk, it can update the chunk with new values. The safe-to-copy bit only matters if the PNG editor does not understand the given chunk. In that case, a PNG editor would not know how to update the values and must either blindly copy the existing chunk (if it is safe) or discard the chunk.

The safe-to-copy bit has existed since the original PNG spec. There are historical clues and implications to help clarify the safe-to-copy bit’s meaning. Since all chunks must use this bit, all official chunks from previous specs also provide clues.

Findings summary

Every chunk is safe-to-copy, unless

1. it depends on some other data outside the chunk, and
2. that other data might be changed.

Perhaps a better name than “safe-to-copy” would be “depends-on-data-elsewhere”.

A “PNG editor” is expected to make changes. And there is no wording to measure how substantive a change may be before the significance of the safe-to-copy bit matters. Rather, the spec clearly says “**any** changes” (emphasis preserved).

The new mDCv, cLLi, and eXIf chunks do depend on the image data. They need to be renamed to mDCV, cLLI, and eXIF, respectively. (Exif contains information about how an image should be rotated for display. If the image data changes, that rotation might no longer apply.)

Safe-to-copy & “PNG editor” definitions

If a chunk does not depend on any other data (such as the tEXt chunk) it is always safe-to-copy. Otherwise, if a chunk depends on other data which is altered the chunk must be updated.

The meaning of the safe-to-copy bit only applies to “PNG editors”. § 5.4 Chunk naming conventions says of the safe-to-copy bit “This property bit … is needed by PNG editors.” Thus, it is important to understand what constitutes a “PNG editor”.

The spec definition of “PNG editor” says “process or device that creates a modification of an existing PNG datastream, preserving unmodified ancillary information wherever possible, and obeying the chunk ordering rules, even for unknown chunk types.”

“Preserving” is key here. An image editor could preserve chunks. Or it could load the image & metadata, discarding where that data came from. Thus not all image editors are PNG editors because not all preserve the PNG information. But some image editors could also be PNG editors. § 14.2 Behavior of PNG editors spells this out, saying “Ordinary image editors are not PNG editors because they usually discard all unrecognized information while reading in an image.”

This also speaks to the extent that a PNG editor might modify a PNG. A PNG editor can make extensive changes, so long as it preserves what it must.

There was an assertion that “PNG editor” means a program which does not make any material changes to the image. For example, a program could apply a better Deflate compression technique, leaving all pixel values identical. However, every chunk is safe-to-copy if a PNG editor leaves all pixel values identical. § 14.2 Behavior of PNG editors says “If a chunk's safe-to-copy bit is 0 (unsafe-to-copy), it indicates that the chunk depends on the image data.”

This means it must be expected that a PNG editor could change image data.

This could leave an interpretation for the definition of “material changes”. For example, changing an image from 16-bit to 8-bit would preserve everything about the image except it introduces color banding. The spec says “If the program has made **any** changes to **critical** chunks, including addition, modification, deletion, or reordering of critical chunks, then unrecognized unsafe chunks shall not be copied to the output PNG datastream” (emphasis preserved).

This shows a PNG editor is expected to change critical chunks to a degree that a dependency on the image data is no longer correct.

If a PNG editor only applied a better Deflate compression technique, the only dependency which could break is something that relies on the Deflate datastream rather than the pixel values. Historically, this possibility had been considered (again, § 14.2 Behavior of PNG editors says “image data”). It is being considered for future editions of the spec, to enable parallel decoding. But that is a separate topic. The spec clearly states “any changes” and does not make any mention about a change being substantive enough.

Possible confusion

§ 14.2 Behavior of PNG editors only makes room for a chunk to depend on data within a critical chunk. It cannot depend on an ancillary chunk. It says “...changes to **critical** chunks” (emphasis preserved). This is echoed in the ordering rules, which say an unsane-to-copy chunk can be reordered so long as it maintains its position relative to any critical chunk.

The intention here is if a chunk depends on data elsewhere (but must be data in a critical chunk), its ordering might also be important relative to the critical chunk. We do not know which critical chunk it depends on. So its order relative to all of them must be preserved.

However, it goes on to say the unsafe-to-copy bit “...indicates that the chunk depends on the image data”. Notice, the spec was clear to call out image data (which is critical) and does not allow a dependency on metadata or anything else.

It is odd to mix “critical chunks” and “image data”. This invites uncertainty and confusion.

There are 4 critical chunks: IHDR, PLTE, IDAT, and IEND. IDAT is named “image data” in the spec. The other chunks are not image data.

IHDR and IEND are critical chunks that are not image data. However, they must come first and last. So ordering relative to them would be preserved anyway. PLTE is palette data.

There are chunks which depend on the palette, not image data. For example, a bKGD chunk specifies the background color using a palette entry.

It could be argued that the palette is “image data”. Except the spec clearly defines “image data” to be a 1D array of scanlines.

Future considerations

It is clear chunks can depend on data other than image data. They already do. Future spec editions should remove the “image data” requirement.

It is possible for a chunk to depend on an ancillary (non-critical) chunk. A hypothetical situation might be bonus content that is gated behind a license key chunk. Future spec editions should consider reworking this.

Future editions might consider explicitly specifying chunk dependency. Currently, a chunk simply indicates that it depends on some other chunk. But that means a change to any chunk would force a purge of unknown unsafe-to-copy chunks. They could instead be preserved if their dependency was unaltered. However, this would require introducing chunks IDs, which is unlikely to happen because there is no backwards-compatible way to add it to existing chunks.

The pHYs chunk is safe-to-copy. However, it arguably shouldn't be. If a PNG editor loads an image but doesn't recognize pHYs, it might alter the image assuming the current physical dimensions (ignoring the original physical dimensions). When saving those changes, the original pHYs data is no longer correct. It should only be preserved if the PNG editor understands the chunk & updates its values. But that is not when the safe-to-copy bit applies.

Notes

A dependency could mean two things. It could mean the meaning of this data depends on some other data (such as the bKGD chunk). Or it could mean this data has an impact on some other data (such as the cHRM chunk).

Notice, those two cases indicate ordering requirements. If this data depends on some other data, it should come after the data it depends on. That enables it to be parsed immediately. Alternatively, if this data impacts some other data, it should come before it.. That also enables immediate parsing.

It is currently not specified which of these two cases are used by a chunk. If ordering rules are reworked later, this information could help.

Conclusion

The meaning of the safe-to-copy bit is that this chunk depends on data elsewhere, outside of the chunk.

If there was a way to indicate what data the chunk depends on, the chunk would be safe to copy unless that data changed. However, that extra information is not currently tracked. Rather, a PNG editor must assume if it changed any data then it may have changed the data this chunk depended on.

An image editor can be a PNG editor. A PNG editor can make major changes. Its only requirement is that it preserve the PNG datastream where aplicable.

There are confusing parts of the spec which should be clarified in a future edition.