PNG-GROUP-DRAFT
PNG-Proposals
Expires: 10 Oct 1996

PNG Development Group
png-list@dworkin.wustl.edu
06 Apr 1996

# PNG Proposed Chunks, draft 0.960406

## File

png-proposed-chunks-0.960406.txt

## Status of this Memo

This document is an informal draft of the PNG development group.

Comments on this document can be sent to the PNG specification maintainers at `png-info@uunet.uu.net` or at `png-list@dworkin.wustl.edu`.

Distribution of this memo is unlimited.

At present, the latest version of this document is available on the World Wide Web from `ftp://swrinde.nde.swri.edu/pub/png-group/documents/`.

## Notices

Copyright © 1996 Thomas Boutell

Permission is granted to copy and distribute this document for any purpose and without charge, provided that the copyright notice and this notice are preserved, and that any substantive changes or deletions from the original are clearly marked.

## Abstract

This document describes some special-purpose chunk types that have been proposed for use in various PNG (Portable Network Graphics) applications. They have not yet been approved for registration by the PNG developers, and therefore are experimental and their format is subject to change. The proposed chunks are `alIG`, `fiNG`, and `spLT`.

## Contents



# 1 Introduction

Chunks described here have been proposed to the PNG development group. Their definitions are subject to change until such time as the group formally registers them. No decoder is required or expected to implement these chunks except for experimental or evaluation purposes. Comments on these proposals, and new proposals for additional chunk types, should be sent to the PNG specification maintainers at `png-info@uunet.uu.net`. The basic PNG specification is available from the W3C archive at `http://www.w3.org/pub/WWW/TR/WD-png`.

# 2 Using Proposed PNG Chunks

No chunks described in this document have yet been registered by the PNG maintainers. Therefore they have a lower-case letter in the second position of the chunkname. They are **experimental** chunks and the format is subject to change. If and when any become registered, the second letter of the chunk name will become uppercase, the chunk description will be moved either to a PNG extensions document or to the core PNG specification, and there will be no further change to the format.

Whenever you use any of these unregistered chunks you should also include a `tEXt` chunk describing it, for example:

```
tEXtComment\0
This file contains a spLT chunk written according to the
format given in Version 0.960406 of the PNG Proposed Chunks document.
```

# 3 Summary of Proposed Chunks

This table summarizes some properties of the proposed chunks described in this document.

```
Chunk name      Multiple   Ordering
                   OK?     constraints

alIG               No      Before IDAT
fiNG               No      None
spLT               Yes     Before IDAT
```

# 4 Proposed Chunk Specifications

This section provides the detailed specifications for the proposed chunks.

## 4.1 `alIG` Alignment

This chunk may be used to provide alignment information to applications. It is especially useful for transparent images in which the edges of the interesting part of the picture do not correspond exactly to the edges of the image data. It can be helpful in aligning the picture with adjacent text, if there is a specific location in the picture that should be aligned with the text baseline. (We use "picture" to mean the thing depicted, as opposed to "image" which is the entire width x height rectangle described by the PNG stream.)

This chunk's contents are

```
left      4 bytes  left side of picture
center    4 bytes  center of picture (horizontal)
right     4 bytes  right side of picture
top       4 bytes  top of picture
middle    4 bytes  middle of picture (vertical)
baseline  4 bytes  typographic baseline of picture (vertical)
bottom    4 bytes  bottom of picture
```

All values are signed 32-bit integer values, measured in pixels downward from the top of the image or rightward from the left edge of the image.

Applications will not normally try to use all of the alignment values at once. A browser might use the "center" value when it wishes to center the image left-to-right on a page. A typographic system might use the "baseline" value to line up an image that contains a fancy capital letter with the baseline of the adjacent text, or the "middle" value to line up images containing mathematical equations. Typographic systems could also determine the typographic "height", "depth", "width", and reference point values:

```
font_height = baseline - top
font_width  = right - left
font_depth  = bottom - baseline
font_ref_x  = left
font_ref_y  = baseline
```

If the `alIG` chunk is not present, applications can assume the following values:

```
left     = 0
center   = width/2
right    = width
top      = 0
middle   = height/2
baseline = 3/4 * height
bottom   = height
```

If an encoder writes the `alIG` chunk, it must supply all of the fields, and should use these values as defaults.

Only one `alIG` chunk is permitted in a PNG datastream. If present, it must appear prior to the first `IDAT` chunk.

## 4.2 `fiNG` Fingerprint

This chunk contains an ADLER-32 fingerprint of the uncompressed, unfiltered, uninterlaced image data, that can be used for rapidly determining whether two files contain the same image, without having to actually decode the IDAT chunks.

The `fiNG` chunk's contents are

```
fingerprint 4 bytes
```

The fingerprint is the ADLER-32 check value computed on the image data, promoted by left-bit-replication (see bit_depth scaling in PNG spec) to 16-bit RGBA (colortype 6). Scaling is done without regard to the `sBIT` chunk, if present. It is calculated without regard to any ancillary chunks. The value is computed on the data in the same order as the bytes would appear in a color type 6 datastream, except that no filter bytes are included in the calculation. The ADLER-32 check value is computed as described in [link to adler-32 code in zlib docs]

An image's fingerprint will not change if the image is changed from interlaced to noninterlaced, or compressed with a different method, or filtered differently, or if any ancillary chunks are added, modified, or removed. It will change if there is any lossy conversion of the pixel data or if the PLTE data is changed in an indexed-color image.

While the `fiNG` chunk does not have any ordering requirements, You should put it early in the file for quick access.

### 4.3 spLT Suggested Palette

This chunk may be used to suggest a reduced palette to be used when the display device is not capable of displaying the full range of colors present in the image. If present, it provides a recommended set of colors, with alpha and frequency information, that may be used to construct a reduced palette to which the truecolor image may be quantized.

This chunk's contents are a zero-byte-terminated text string that names the palette, followed by a series of palette entries, each a ten-byte series, containing five unsigned integers:

```
name:      n bytes (ASCII text)
null byte  1 byte  (terminator)

red:       2 bytes (0 = black, 65535 = red)
green:     2 bytes (0 = black, 65535 = green)
blue:      2 bytes (0 = black, 65535 = blue)
alpha:     2 bytes (0 = fully transparent, 65535 = fully opaque)
frequency: 2 bytes (relative frequency of occurrence)
...
```

There can be any number of entries; a decoder determines the number of entries from the remaining chunk length after the null-terminated "name" string. This length not divisible by ten is an error. Entries must appear in decreasing order of "frequency".

The "name" (e.g. "rgba512 8-8-4-2 color cube", "rgb256 winter scenery", "rgb242 6-6-6 color cube plus 26 gray levels", "Windows white background", "50-color rgb palette for use with early versions of Mosaic") identifies the palette, which may permit applications or people to choose the appropriate one when more than one suggested palette appears in a PNG file. The "name" string must consist only of printable ASCII characters and may not have leading or trailing blanks, but may have single embedded blanks. There must be at least one and no more than 79 characters in the name. Names are case-sensitive.

The red, green, and blue values are not premultiplied by alpha, nor are they precomposited against any background. A decoder can build a palette by compositing those palette entries against any background color or set of background colors that it chooses. See [link to bKGD]

Each frequency entry is proportional to the fraction of pixels in the image that are closest to that palette entry, without regard to any compositing against a background palette. The exact scale factor is chosen by the encoder, but should be chosen so that the range of individual values reasonably fills the range 0 to 65535. It is acceptable to artificially inflate the "frequency" values for "important" colors such as those in a company logo or in the facial features of a portrait. Zero is a valid value for frequency.

Note that the palette uses 16 bits (2 bytes) per value regardless of the image bit depth specification. Decoders wishing to construct 8-bit palettes can accomplish this by scaling down the RGB entries to 8 bits. [link to bit_depth rescaling].

Note: Earlier versions of the PNG specification recommended that the `PLTE` [link to PLTE] and `hIST` chunks be used for this purpose. While this is still allowed, to maintain backward compatibility, the `spLT` chunk is preferable, particularly when transparent pixels are present. When both the `PLTE` and `spLT` chunks are present, the `PLTE` data should only be used for decoding the indexed-color (color type 3) pixels, and the `spLT` data should be used for constructing the display palette. If the `hIST` chunk is also present, decoders that process the `spLT` chunk should ignore it.

This chunk may appear for any color type. If this chunk does appear, it must precede the first `IDAT` chunk. There may be multiple `spLT` chunks, with different names.

# 5 Chunks not described here

Additional chunks have been proposed for scientific visualization purposes. Since they are not expected to be implemented by general-purpose decoders, the are described separately in the PNG Scientific Visualization Chunks document. The proposed chunks described there are `drNG/DrNG`, `faLS`, `loGE/LoGE`, `xySC`, and `zsCL`.

# 6 Security considerations

Security considerations are addressed in the basic PNG specification, `http://www.w3.org/pub/WWW/TR/WD-png`.

The same precautions taken when displaying `tEXt` data should be taken when displaying the text contained in the "name" strings of the `spLT` chunk. Viewers should not display these strings directly without first checking for the presence of nonprintable characters, and for the <ESC> character in particular.

No known additional security hazards are posed by the chunks described here.

# 7 Credits

### Trademarks

- Windows is a trademark of MicroSoft Corp.

### Contributors

Contributors' names are presented in alphabetical order:

- Mark Adler, madler@alumni.caltech.edu

- Thomas Boutell, boutell@boutell.com
- Christian Brunschen, cb@df.lth.se
- Adam M. Costello, amc@cs.wustl.edu
- Lee Daniel Crocker, lee@piclab.com
- Andreas Dilger, adilger@enel.ucalgary.ca
- Oliver Fromme, fromme@rz.tu-clausthal.de
- Todd French, todd@lfd.physics.uiuc.edu
- Jean-loup Gailly, gzip@prep.ai.mit.edu
- Chris Herborth, chrish@qnx.com
- Alex Jakulin, alex@hermes.si
- Neal Kettler, kettler@cs.colostate.edu
- Jim King, jimk@mathtype.com
- Tom Lane, tgl@sss.pgh.pa.us
- Alexander Lehmann, alex@hal.rhein-main.de
- Chris Lilley, chris.lilley@mcc.ac.uk
- Dave Martindale, davem@cs.ubc.ca
- Owen Mortensen, ojm@csi.compuserve.com
- Robert P. Poole, lionboy@primenet.com
- Glenn Randers-Pehrson, glennrp@arl.mil or randeg@alumni.rpi.edu
- Greg Roelofs, newt@uchicago.edu
- Willem van Schaik, gwillem@ntuvax.ntu.ac.sg
- Guy Schalnat, schalnat@group42.com
- Paul Schmidt, pschmidt@photodex.com
- Tim Wegner, 71320.675@compuserve.com


## Editor


- Glenn Randers-Pehrson, glennrp@arl.mil or randeg@alumni.rpi.edu


*End of PNG Proposed Chunks Listing*