Portable Network Graphics

Portable Network Graphics (**PNG**, officially pronounced /pɪŋ/[2][3] *PING*, colloquially pronounced /ˌpiːɛnˈdʒiː/[4] *PEE-en-JEE*) is a raster-graphics file format that supports lossless data compression.[5] PNG was developed as an improved, non-patented replacement for Graphics Interchange Format (GIF) — unofficially, the initials PNG stood for the recursive acronym "PNG's not GIF".[6]

PNG supports palette-based images (with palettes of 24-bit RGB or 32-bit RGBA colors), grayscale images (with or without an alpha channel for transparency), and full-color non-palette-based RGB or RGBA images. The PNG working group designed the format for transferring images on the Internet, not for professional-quality print graphics; therefore non-RGB color spaces such as CMYK are not supported. A PNG file contains a single image in an extensible structure of *chunks*, encoding the basic pixels and other information such as textual comments and integrity checks documented in RFC 2083.[7]

PNG files use the file extension `PNG` or `png` and have been assigned the MIME media type `image/png`.[8] PNG was published as informational RFC 2083 in March 1997 and as an ISO/IEC 15948 standard in 2004.[1]

Contents

- History and development
- PNG Working Group
- File format
  - File header
  - "Chunks" within the file
    - Critical chunks
    - Ancillary chunks
  - Pixel format
  - Transparency of image
  - Compression
    - Filtering
  - Interlacing
  - Animation

Portable Network Graphics

A PNG image with an 8-bit transparency channel, overlaid onto a checkered background, typically used in graphics software to indicate transparency

Filename extension	`.png`
Internet media type	`image/png`
Type code	PNGf PNG
Uniform Type Identifier (UTI)	public.png
UTI conformation	public.image
Magic number	`89 50 4e 47 0d 0a 1a 0a` (8 bytes Hexadecimal)
Developed by	PNG Development Group (donated to W3C)
Initial release	1 October 1996
Type of format	Lossless bitmap image format
Extended to	APNG, JNG and MNG
Standard	ISO/IEC 15948,[1] IETF RFC 2083
Open format?	Yes

- Examples
- **Advantages**
- **Comparison with other file formats**
  - Graphics Interchange Format (GIF)
  - JPEG
  - JPEG-LS
  - TIFF
- **Software support**
  - Bitmap graphics editor support for PNG
  - Web browser support for PNG
  - Operating system support for PNG icons
- **File size and optimization software**
  - Compared to GIF
  - File size factors
    - Lossy PNG compression
  - Image editing software
  - Optimizing tools
    - Tool list
    - Ancillary chunk removal
    - Filter optimization
    - DEFLATE optimization
  - Icon optimization
- **See also**
- **Explanatory notes**
- **References**
- **Further reading**
- **External links**

History and development

The motivation for creating the PNG format was the realization that, on 28 December 1994, the Lempel–Ziv–Welch (LZW) data compression algorithm used in the Graphics Interchange Format (GIF) format was patented by Unisys. The patent required that all software supporting GIF pay royalties, leading to a flurry of criticism from Usenet users. One of them was Thomas Boutell, who on 4 January 1995 posted a precursory discussion thread on the Usenet newsgroup "comp.graphics" in which he devised a plan for a free alternative to GIF. Other users in that thread put forth many propositions that would later be part of the final file format. Oliver Fromme, author of the popular JPEG viewer QPEG, proposed the PING name, eventually becoming PNG, a recursive acronym meaning *PING is not GIF*, and also the `.png` extension. Other suggestions later implemented included the deflate compression algorithm and 24-bit color support, the lack of the latter in GIF also motivating the team to create their file format. The group would become known as the PNG Development Group, and as the discussion rapidly expanded, it later used a mailing list associated with a CompuServe forum.[2][9]

The full specification of PNG was released under the approval of W3C on 1 October 1996, and later as RFC 2083 on 15 January 1997. The specification was revised on 31 December 1998 as version 1.1, which addressed technical problems for gamma and color correction. Version 1.2, released on 11 August 1999, added the `iTXt` chunk as the specification's only change, and a reformatted version of 1.2 was released as a second edition of the W3C standard on 10 November 2003,[10] and as an International Standard (ISO/IEC 15948:2004 (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=29581&scopelist=PROGRAMME)) on 3 March 2004.[11][1]

Although GIF allows for animation, it was decided that PNG should be a single-image format.[12] In 2001, the developers of PNG published the Multiple-image Network Graphics (MNG) format, with support for animation. MNG achieved moderate application support, but not enough among mainstream web browsers and no usage among web site designers or publishers. In 2008, certain Mozilla developers published the Animated Portable Network Graphics (APNG) format with similar goals. APNG is a format that is natively supported by Gecko- and Presto-based web browsers and is also commonly used for thumbnails on Sony's PlayStation Portable system (using the normal PNG file extension). In 2017, Chromium based browsers adopted APNG support. In January 2020, Microsoft Edge became Chromium based, thus inheriting support for APNG. With this all major browsers now support APNG.

PNG Working Group

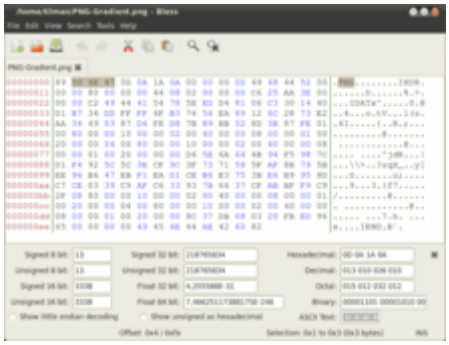
The original PNG specification was authored by an ad hoc group of computer graphics experts and enthusiasts. Discussions and decisions about the format were conducted by email. The original authors listed on RFC 2083 are:[13]

- Editor: Thomas Boutell
- Contributing Editor: Tom Lane
- Authors (in alphabetical order by last name): Mark Adler, Thomas Boutell, Christian Brunschen, Adam M. Costello, Lee Daniel Crocker, Andreas Dilger, Oliver Fromme, Jean-loup Gailly, Chris Herborth, Aleks Jakulin, Neal Kettler, Tom Lane, Alexander Lehmann, Chris Lilley, Dave Martindale, Owen Mortensen, Keith S. Pickens, Robert P. Poole, Glenn Randers-Pehrson, Greg Roelofs, Willem van Schaik, Guy Schalnat, Paul Schmidt, Tim Wegner, Jeremy Wohl

File format

File header

A PNG file starts with an 8-byte signature[14] (refer to hex editor image on the right):



The PNG image viewed with a hex editor application for Ubuntu.

Values (hex)	Purpose
`89`	Has the high bit set to detect transmission systems that do not support 8-bit data and to reduce the chance that a text file is mistakenly interpreted as a PNG, or vice versa.
`50 4E 47`	In ASCII, the letters *PNG*, allowing a person to identify the format easily if it is viewed in a text editor.
`0D 0A`	A DOS-style line ending (CRLF) to detect DOS-Unix line ending conversion of the data.
`1A`	A byte that stops display of the file under DOS when the command type has been used—the end-of-file character.
`0A`	A Unix-style line ending (LF) to detect Unix-DOS line ending conversion.

"Chunks" within the file

After the header, comes a series of chunks,[15] each of which conveys certain information about the image. Chunks declare themselves as *critical* or *ancillary*, and a program encountering an ancillary chunk that it does not understand can safely ignore it. This chunk-based storage layer structure, similar in concept to a container format or to Amiga's IFF, is designed to allow the PNG format to be extended while maintaining compatibility with older versions—it provides forward compatibility, and this same file structure (with different signature and chunks) is used in the associated MNG, JNG, and APNG formats.

A chunk consists of four parts: length (4 bytes,[16] big-endian), chunk type/name (4 bytes[17]), chunk data (length bytes) and CRC (cyclic redundancy code/checksum; 4 bytes[16]). The CRC is a network-byte-order CRC-32 computed over the chunk type and chunk data, but not the length.

Length	Chunk type	Chunk data	CRC
4 bytes	4 bytes	*Length* bytes	4 bytes

Chunk types are given a four-letter case sensitive ASCII type/name; compare FourCC. The case of the different letters in the name (bit 5 of the numeric value of the character) is a bit field that provides the decoder with some information on the nature of chunks it does not recognize.

The case of the first letter indicates whether the chunk is critical or not. If the first letter is uppercase, the chunk is critical; if not, the chunk is ancillary. Critical chunks contain information that is necessary to read the file. If a decoder encounters a critical chunk it does not recognize, it must abort reading the file or supply the user with an appropriate warning.

The case of the second letter indicates whether the chunk is "public" (either in the specification or the registry of special-purpose public chunks) or "private" (not standardised). Uppercase is public and lowercase is private. This ensures that public and private chunk names can never conflict with each other (although two private chunk names could conflict).

The third letter must be uppercase to conform to the PNG specification. It is reserved for future expansion. Decoders should treat a chunk with a lower case third letter the same as any other unrecognised chunk.

The case of the fourth letter indicates whether the chunk is safe to copy by editors that do not recognize it. If lowercase, the chunk may be safely copied regardless of the extent of modifications to the file. If uppercase, it may only be copied if the modifications have not touched any critical chunks.

Critical chunks

A decoder must be able to interpret critical chunks to read and render a PNG file.

- `IHDR` must be the first chunk; it contains (in this order) the image's
  - width (4 bytes)
  - height (4 bytes)
  - bit depth (1 byte, values 1, 2, 4, 8, or 16)
  - color type (1 byte, values 0, 2, 3, 4, or 6)
  - compression method (1 byte, value 0)
  - filter method (1 byte, value 0)
  - interlace method (1 byte, values 0 "no interlace" or 1 "Adam7 interlace") (13 data bytes total).[10]

As stated in the World Wide Web Consortium, bit depth is defined as "the number of bits per sample or per palette index (not per pixel)".[10]

- `PLTE` contains the palette: a list of colors.
- `IDAT` contains the image, which may be split among multiple IDAT chunks. Such splitting increases filesize slightly, but makes it possible to generate a PNG in a streaming manner. The IDAT chunk contains the actual image data, which is the output stream of the compression algorithm.[18]
- `IEND` marks the image end; the data field of the IEND chunk has 0 bytes/is empty.[19]

The `PLTE` chunk is essential for color type 3 (indexed color). It is optional for color types 2 and 6 (truecolor and truecolor with alpha) and it must not appear for color types 0 and 4 (grayscale and grayscale with alpha).

Ancillary chunks

Other image attributes that can be stored in PNG files include gamma values, background color, and textual metadata information. PNG also supports color management through the inclusion of ICC color profiles.[20]

- `bKGD` gives the default background color. It is intended for use when there is no better choice available, such as in standalone image viewers (but not web browsers; see below for more details).
- `cHRM` gives the chromaticity coordinates of the display primaries and white point.
- `dSIG` is for storing digital signatures.[21]
- `eXIf` stores Exif metadata.[22]
- `gAMA` specifies gamma. The gAMA chunk contains only 4 bytes, and its value represents the gamma value multiplied by 100,000; for example, the gamma value 1/3.4 calculates to 29411.7647059 ((1/3.4)*(100,000)) and is converted to an integer (29412) for storage.[23]
- `hIST` can store the histogram, or total amount of each color in the image.
- `iCCP` is an ICC color profile.
- `iTXt` contains a keyword and UTF-8 text, with encodings for possible compression and translations marked with language tag. The Extensible Metadata Platform (XMP) uses this chunk with a keyword 'XML:com.adobe.xmp'
- `pHYs` holds the intended pixel size (or pixel aspect ratio); the pHYs contains "Pixels per unit, X axis" (4 bytes), "Pixels per unit, Y axis" (4 bytes), and "Unit specifier" (1 byte) for a total of

9 bytes.[24]

- `sBIT` (significant bits) indicates the color-accuracy of the source data; this chunk contains a total of between 1 and 5 bytes, depending on the color type.[25][26][27]
- `sPLT` suggests a palette to use if the full range of colors is unavailable.
- `sRGB` indicates that the standard sRGB color space is used; the sRGB chunk contains only 1 byte, which is used for "rendering intent" (4 values—0, 1, 2, and 3—are defined for rendering intent).[28]
- `sTER` stereo-image indicator chunk for stereoscopic images.[29]
- `tEXt` can store text that can be represented in ISO/IEC 8859-1, with one key-value pair for each chunk. The "key" must be between 1 and 79 characters long. Separator is a null character. The "value" can be any length, including zero up to the maximum permissible chunk size minus the length of the keyword and separator. Neither "key" nor "value" can contain null character. Leading or trailing spaces are also disallowed.
- `tIME` stores the time that the image was last changed.
- `tRNS` contains transparency information. For indexed images, it stores alpha channel values for one or more palette entries. For truecolor and grayscale images, it stores a single pixel value that is to be regarded as fully transparent.
- `zTXt` contains compressed text (and a compression method marker) with the same limits as `tEXt`.

The lowercase first letter in these chunks indicates that they are not needed for the PNG specification. The lowercase last letter in some chunks indicates that they are safe to copy, even if the application concerned does not understand them.

Pixel format

Pixels in PNG images are numbers that may be either indices of sample data in the palette or the sample data itself. The palette is a separate table contained in the PLTE chunk. Sample data for a single pixel consists of a tuple of between one and four numbers. Whether the pixel data represents palette indices or explicit sample values, the numbers are referred to as channels and every number in the image is encoded with an identical format.

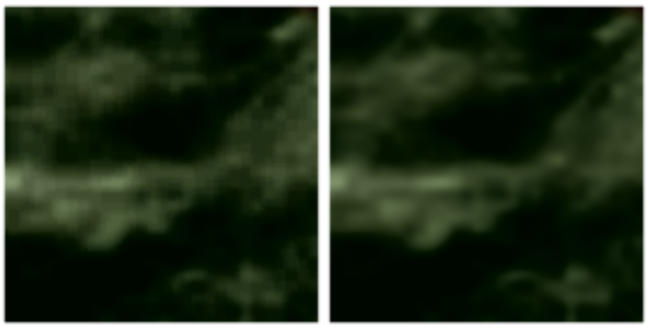
PNG color type[10]

Color type	Channels	Bits per channel				
		1	**2**	**4**	**8**	**16**
Indexed	**1**	1	2	4	8	
Grayscale	**1**	1	2	4	8	16
Grayscale and alpha	**2**				16	32
Truecolor	**3**				24	48
Truecolor and alpha	**4**				32	64

The permitted formats encode each number as an unsigned integer value using a fixed number of bits, referred to in the PNG specification as the *bit depth*. Notice that this is not the same as color depth, which is commonly used to refer to the total number of bits in each pixel, not each channel. The permitted bit depths are summarized in the table along with the total number of bits used for each pixel.

The number of channels depends on whether the image is grayscale or color and whether it has an alpha channel. PNG allows the following combinations of channels, called the *color type*.

0 (000_2)	grayscale
2 (010_2)	red, green and blue: rgb/truecolor
3 (011_2)	indexed: channel containing indices into a palette of colors
4 (100_2)	grayscale and alpha: level of opacity for each pixel
6 (110_2)	red, green, blue and alpha



A demonstration of the colour depth in a PNG file, in bits per channel. Left: 8 bits, Right: 16 bits. Note the artifacts, adjusted contrast for clarity.

The color type is specified as an 8-bit value however only the low 3 bits are used and, even then, only the five combinations listed above are permitted. So long as the color type is valid it can be considered as a bit field as summarized in the adjacent table:

- bit value 1: the image data stores palette indices. This is only valid in combination with bit value 2;
- bit value 2: the image samples contain three channels of data encoding trichromatic colors, otherwise the image samples contain one channel of data encoding relative luminance,
- bit value 4: the image samples also contain an alpha channel expressed as a linear measure of the opacity of the pixel. This is not valid in combination with bit value 1.

PNG color types

Color type	Name	Binary				Masks
			A	C	P	
0	**Grayscale**	0	0	0	0	
2	**Truecolor**	0	0	1	0	color
3	**Indexed**	0	0	1	1	color, palette
4	**Grayscale and alpha**	0	1	0	0	alpha
6	**Truecolor and alpha**	0	1	1	0	alpha, color

With indexed color images, the palette always stores trichromatic colors at a depth of 8 bits per channel (24 bits per palette entry). Additionally, an optional list of 8-bit alpha values for the palette entries may be included; if not included, or if shorter than the palette, the remaining palette entries are assumed to be opaque. The palette must not have more entries than the image bit depth allows for, but it may have fewer (for example, if an image with 8-bit pixels only uses 90 colors then it does not need palette entries for all 256 colors). The palette must contain entries for all the pixel values present in the image.

The standard allows indexed color PNGs to have 1, 2, 4 or 8 bits per pixel; grayscale images with no alpha channel may have 1, 2, 4, 8 or 16 bits per pixel. Everything else uses a bit depth per channel of either 8 or 16. The combinations this allows are given in the table above. The standard requires that decoders can read all supported color formats, but many image editors can only produce a small subset of them.

Transparency of image

PNG offers a variety of transparency options. With true-color and grayscale images either a single pixel value can be declared as transparent or an alpha channel can be added (enabling any percentage of partial transparency to be used). For paletted images, alpha values can be added to palette entries. The number of such values stored may be less than the total number of palette entries, in which case the remaining entries are considered fully opaque.

The scanning of pixel values for binary transparency is supposed to be performed before any color reduction to avoid pixels becoming unintentionally transparent. This is most likely to pose an issue for systems that can decode 16-bits-per-channel images (as is required for compliance with the specification) but only output at 8 bits per channel (the norm for all but the highest end systems).

Alpha *storage* can be "associated" ("premultiplied") or "unassociated", but PNG standardized[30] on "unassociated" ("non-premultiplied") alpha, which means that imagery is not alpha *encoded*; the emissions represented in RGB are not the emissions at the pixel level. This means that the over operation will multiply the RGB emissions by the alpha, and cannot represent emission and occlusion properly.

Compression

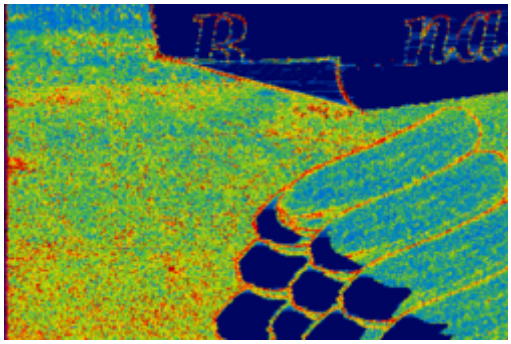
PNG uses a 2-stage compression process:

- pre-compression: filtering (prediction)
- compression: DEFLATE

PNG uses DEFLATE, a non-patented lossless data compression algorithm involving a combination of LZ77 and Huffman coding. Permissively-licensed DEFLATE implementations, such as zlib, are widely available.

Example with several types of image content

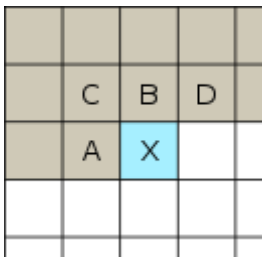
Compared to formats with lossy compression such as JPEG, choosing a compression setting higher than average delays processing, but often does not result in a significantly smaller file size.


Representation of bit cost per pixel for above PNG file (red=expensive, blue=cheap)

Filtering

Before DEFLATE is applied, the data is transformed via a prediction method: a single *filter method* is used for the entire image, while for each image line, a *filter type* is chosen to transform the data to make it more efficiently compressible.[31] The filter type used for a scanline is prepended to the scanline to enable inline decompression.

There is only one filter method in the current PNG specification (denoted method 0), and thus in practice the only choice is which filter type to apply to each line. For this method, the filter predicts the value of each pixel based on the values of previous neighboring pixels, and subtracts the predicted color of the pixel from the actual value, as in DPCM. An image line filtered in this way is often more compressible than the raw image line would be, especially if it is similar to the line above, since the differences from prediction will generally be clustered around 0, rather than spread over all possible image values. This is particularly important in relating separate rows, since DEFLATE has no understanding that an image is a 2D entity, and instead just sees the image data as a stream of bytes.

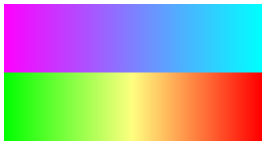


PNG's filter method 0 can use the data in pixels A, B, and C to predict the value for X.

There are five filter types for filter method 0; each type predicts the value of each byte (of the image data before filtering) based on the corresponding byte of the pixel to the left (*A*), the pixel above (*B*), and the pixel above and to the left (*C*) or

some combination thereof, and encodes the *difference* between the predicted value and the actual value. Filters are applied to byte values, not pixels; pixel values may be one or two bytes, or several values per byte, but never cross byte boundaries. The filter types are:[32]

Type byte	Filter name	Predicted value
0	None	Zero (so that the raw byte value passes through unaltered)
1	Sub	Byte *A* (to the left)
2	Up	Byte *B* (above)
3	Average	Mean of bytes *A* and *B*, rounded down
4	Paeth	*A*, *B*, or *C*, whichever is closest to $p = A + B - C$



A PNG with 256 colors, which is only 251 bytes large with pre-filter. The same image as a GIF would be more than thirteen times larger.

The Paeth filter is based on an algorithm by Alan W. Paeth.[33] Compare to the version of DPCM used in lossless JPEG, and to the discrete wavelet transform using 1×2, 2×1, or (for the Paeth predictor) 2×2 windows and Haar wavelets.

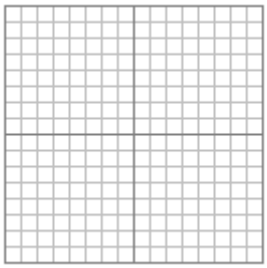
Compression is further improved by choosing filter types adaptively on a line-by-line basis. This improvement, and a heuristic method of implementing it commonly used by PNG-writing software, were created by Lee Daniel Crocker, who tested the methods on many images during the creation of the format;[34] the choice of filter is a component of file size optimization, as discussed below.

If interlacing is used, each stage of the interlacing is filtered separately, meaning that the image can be progressively rendered as each stage is received; however, interlacing generally makes compression less effective.

Interlacing

PNG offers an optional 2-dimensional, 7-pass interlacing scheme—the Adam7 algorithm. This is more sophisticated than GIF's 1-dimensional, 4-pass scheme, and allows a clearer low-resolution image to be visible earlier in the transfer, particularly if interpolation algorithms such as bicubic interpolation are used.[35]

However, the 7-pass scheme tends to reduce the data's compressibility more than simpler schemes.



An illustration of Adam7 interlacing over a 16×16 image.

Animation

PNG itself does not support animation. MNG is an extension to PNG that does; it was designed by members of the PNG Group. MNG shares PNG's basic structure and chunks, but it is significantly more complex and has a different file signature, which automatically renders it incompatible with standard PNG decoders. This means that most web browsers and applications either never supported MNG or dropped support for it.

The complexity of MNG led to the proposal of APNG by developers at the Mozilla Foundation. It is based on PNG, supports animation and is simpler than MNG. APNG offers fallback to single-image display for PNG decoders that do not support APNG. Today, the APNG format is supported by all major web browsers.[36] APNG is supported in Firefox 3.0 and up, Pale Moon (all versions), and Safari 8.0 and

up.[37] Chromium 59.0 added APNG support,[38][39] followed by Google Chrome. Opera supported APNG in versions 10–12.1, but support lapsed in version 15 when it switched to the Blink rendering engine; support was re-added in Opera 46 (inherited from Chromium 59).[40] Microsoft Edge has supported APNG since version 79.0, when it switched to a Chromium-based engine.

The PNG Group decided in April 2007 not to embrace APNG.[41] Several alternatives were under discussion, including ANG, aNIM/mPNG, "PNG in GIF" and its subset "RGBA in GIF".[42] However, currently only APNG has widespread support.

An APNG (animated PNG) file (displays as static image in some web browsers)

Examples

Structure of a very simple PNG file

`89 50 4E 47 0D 0A 1A 0A` PNG signature	`IHDR` Image header	`IDAT` Image data	`IEND` Image end

Contents of a minimal PNG file representing one red pixel

Hex	As characters

`89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52`
`00 00 00 01 00 00 00 01 08 02 00 00 00 90 77 53`
`DE 00 00 00 0C 49 44 41 54 08 D7 63 F8 CF C0 00`
`00 03 01 01 00 18 DD 8D B0 00 00 00 00 49 45 4E`
`44 AE 42 60 82`
	`.PNG....IHDR`
	`..........wS`
	`....IDAT..c....`
	`.....IEN`
	`D.B``

IHDR Chunk

Offset into chunk	Hex Value	Decimal Value	Text	Meaning
0	0x0D	13		IHDR chunk has 13 bytes of content
4	0x49484452		IHDR	Identifies a Header chunk
8	0x01	1		Image is 1 pixel wide
12	0x01	1		Image is 1 pixel high
16	0x08	8		8 bits per pixel (per channel)
17	0x02	2		Color type 2 (RGB/truecolor)
18	0x00	0		Compression method 0 (only accepted value)
19	0x00	0		Filter method 0 (only accepted value)
20	0x00	0		Not interlaced
21	0x907753DE			CRC of chunk's type and content (but not length)

IDAT Chunk

Offset into chunk	Hex Value	Meaning
0	0x0C	IDAT chunk has 12 bytes of content
4	0x49444154	Identifies a Data chunk
8	0x08	DEFLATE compression method using a 256-byte window[43]
9	0xD7	ZLIB FCHECK value, no dictionary used, maximum compression algorithm[43]
10	0x63F8CFC00000	A compressed DEFLATE block using the static Huffman code that decodes to 0x00 0xFF 0x00 0x00[44]
16	0x03010100	The ZLIB check value: the Adler32 checksum of the uncompressed data[43]
20	0x18DD8DB0	CRC of chunk's type and content (but not length)

Displayed in the fashion of hex editors, with on the left side byte values shown in hex format, and on the right side their equivalent characters from ISO-8859-1 with unrecognized and control characters replaced with periods. Additionally the PNG signature and individual chunks are marked with colors. Note they are easy to identify because of their human readable type names (in this example PNG, IHDR, IDAT, and IEND).

Advantages

Reasons to use this International Standard may be:

- **Portability**: Transmission is independent of the software and hardware platform.
- **Completeness**: it's possible to represent truecolor, indexed-color, and greyscale images.
- **Coding and decoding in series**: allows to generate and read data streams in series, that is, the format of the data stream is used for the generation and visualization of images at the moment through serial communication.
- **Progressive presentation**: to be able to transmit data flows that are initially an approximation of the entire image and progressively they improve as the data flow is received.
- **Soundness to transmission errors**: detects the transmission errors of the data stream correctly.
- **Losslessness**: No loss: filtering and compression preserve all information.
- **Efficiency**: any progressive image presentation, compression and filtering seeks efficient decoding and presentation.
- **Compression**: images can be compressed efficiently and consistently.
- **Easiness**: the implementation of the standard is easy.
- **Interchangeability**: any PNG decoder that follows the standards can read all PNG data streams.
- **Flexibility**: allows future extensions and private additions without affecting the previous point.
- **Freedom of legal restrictions**: the algorithms used are free and accessible.

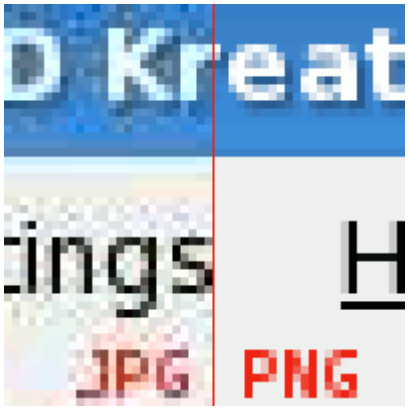
Comparison with other file formats

Graphics Interchange Format (GIF)

- On small images, GIF can achieve greater compression than PNG (see the section on filesize, below).
- On most images, except for the above case, a GIF file has a larger size than an indexed PNG image.
- PNG gives a much wider range of transparency options than GIF, including alpha channel transparency.
- Whereas GIF is limited to 8-bit indexed color, PNG gives a much wider range of color depths, including 24-bit (8 bits per channel) and 48-bit (16 bits per channel) truecolor, allowing for greater color precision, smoother fades, etc.[45] When an alpha channel is added, up to 64 bits per pixel (before compression) are possible.
- When converting an image from the PNG format to GIF, the image quality may suffer due to posterization if the PNG image has more than 256 colors.
- GIF intrinsically supports animated images. PNG supports animation only via unofficial extensions (see the section on animation, above).

PNG images are less widely supported by older browsers. In particular, IE6 has limited support for PNG.[46]

JPEG



Composite image comparing lossy compression in JPEG with lossless compression in PNG: the JPEG artifacts can be easily visible in the background of this kind of image data, where the PNG image has solid color.

The JPEG (Joint Photographic Experts Group) format can produce a smaller file than PNG for photographic (and photo-like) images, since JPEG uses a lossy encoding method specifically designed for photographic image data, which is typically dominated by soft, low-contrast transitions, and an amount of noise or similar irregular structures. Using PNG instead of a high-quality JPEG for such images would result in a large increase in filesize with negligible gain in quality. In comparison, when storing images that contain text, line art, or graphics – images with sharp transitions and large areas of solid color – the PNG format can compress image data more than JPEG can. Additionally, PNG is lossless, while JPEG produces visual artifacts around high-contrast areas. (Such artifacts depend on the settings used in the JPG compression; they can be quite noticeable when a low-quality [high-compression] setting is used.) Where an image contains both sharp transitions and photographic parts, a choice must be made between the two effects. JPEG does not support transparency.

JPEG's lossy compression also suffers from generation loss, where repeatedly decoding and re-encoding an image to save it again causes a loss of information each time, degrading the image. Because PNG is lossless, it is suitable for storing images to be edited. While PNG is reasonably efficient when compressing photographic images, there are lossless compression formats designed specifically for photographic images, lossless WebP and Adobe DNG (digital negative) for example. However these formats are either not widely supported, or are proprietary. An image can be stored losslessly and converted to JPEG format only for distribution, so that there is no generation loss.

While the PNG specification does not explicitly include a standard for embedding Exif image data from sources such as digital cameras, the preferred method for embedding EXIF data in a PNG is to use the non-critical ancillary chunk label `eXIf`.[47]

Early web browsers did not support PNG images; JPEG and GIF were the main image formats. JPEG was commonly used when exporting images containing gradients for web pages, because of GIF's limited color depth. However, JPEG compression causes a gradient to blur slightly. A PNG format reproduces a gradient as accurately as possible for a given bit depth, while keeping the file size small. PNG became the optimal choice for small gradient images as web browser support for the format improved. No images at all are needed to display gradients in modern browsers, as gradients can be created using CSS.

JPEG-LS

JPEG-LS is an image format by the Joint Photographic Experts Group, though far less widely known and supported than the other lossy JPEG format discussed above. It is directly comparable with PNG, and has a standard set of test images.[48] On the Waterloo Repertoire ColorSet, a standard set of test images (unrelated to the JPEG-LS conformance test set), JPEG-LS generally performs better than PNG, by 10–15%, but on some images PNG performs substantially better, on the order of 50–75%.[49] Thus, if both of these formats are options and file size is an important criterion, they should both be considered, depending on the image.

TIFF

Tagged Image File Format (TIFF) is a format that incorporates an extremely wide range of options. While this makes TIFF useful as a generic format for interchange between professional image editing applications, it makes adding support for it to applications a much bigger task and so it has little support in applications not concerned with image manipulation (such as web browsers). The high level of extensibility also means that most applications provide only a subset of possible features, potentially creating user confusion and compatibility issues.

The most common general-purpose, lossless compression algorithm used with TIFF is Lempel–Ziv–Welch (LZW). This compression technique, also used in GIF, was covered by patents until 2003. TIFF also supports the compression algorithm PNG uses (i.e. Compression Tag 0008_{16} 'Adobe-style') with medium usage and support by applications. TIFF also offers special-purpose lossless compression algorithms like CCITT Group IV, which can compress bilevel images (e.g., faxes or black-and-white text) better than PNG's compression algorithm.

PNG supports non-premultiplied alpha only[30] whereas TIFF also supports "associated" (premultiplied) alpha.

Software support

The official reference implementation of the PNG format is the programming library *libpng*.[50] It is published as free software under the terms of a permissive free software license. Therefore, it is usually found as an important system library in free operating systems.

Bitmap graphics editor support for PNG

The PNG format is widely supported by graphics programs, including Adobe Photoshop, Corel's Photo-Paint and Paint Shop Pro, the GIMP, GraphicConverter, Helicon Filter, ImageMagick, Inkscape, IrfanView, Pixel image editor, Paint.NET and Xara Photo & Graphic Designer and many others. Some

programs bundled with popular operating systems which support PNG include Microsoft's Paint and Apple's Photos/iPhoto and Preview, with the GIMP also often being bundled with popular Linux distributions.

Adobe Fireworks (formerly by Macromedia) uses PNG as its native file format, allowing other image editors and preview utilities to view the flattened image. However, Fireworks by default also stores metadata for layers, animation, vector data, text and effects. Such files should not be distributed directly. Fireworks can instead export the image as an optimized PNG without the extra metadata for use on web pages, etc.

Web browser support for PNG

PNG support first appeared in 1997, in Internet Explorer 4.0b1 (32-bit only for NT), and in Netscape 4.04.[51]

Despite calls by the Free Software Foundation[52] and the World Wide Web Consortium (W3C),[53] tools such as gif2png,[54] and campaigns such as Burn All GIFs,[55] PNG adoption on websites was fairly slow due to late and buggy support in Internet Explorer, particularly regarding transparency.[56]

PNG compatible browsers include: Apple Safari, Google Chrome, Mozilla Firefox, Opera, Camino, Internet Explorer 7 (still numerous issues),[57] Internet Explorer 8 (still some issues), Internet Explorer 9 and many others. For the complete comparison, see Comparison of web browsers (Image format support).

Especially versions of Internet Explorer (Windows) below 9.0 (released 2011) have numerous problems which prevent it from correctly rendering PNG images.[57]

- 4.0 crashes on large PNG chunks.[58]
- 4.0 does not include the functionality to view .png files,[59] but there is a registry fix.[57]
- 5.0 and 5.01 have broken OBJECT support.[60]
- 5.01 prints palette images with black (or dark gray) backgrounds under Windows 98, sometimes with radically altered colors.[61]
- 6.0 fails to display PNG images of 4097 or 4098 bytes in size.[62]
- 6.0 cannot open a PNG file that contains one or more zero-length IDAT chunks. This issue was first fixed in security update 947864 (MS08-024). For more information, see this article in the Microsoft Knowledge Base: 947864 (http://support.microsoft.com/kb/947864/) MS08-024: Cumulative Security Update for Internet Explorer.[63]
- 6.0 sometimes completely loses ability to display PNGs, but there are various fixes.[64]
- 6.0 and below have broken alpha-channel transparency support (will display the default background color instead).[65][66][67]
- 7.0 and below cannot combine 8-bit alpha transparency AND element opacity (CSS – filter: Alpha (opacity=xx)) without filling partially transparent sections with black.[68]
- 8.0 and below have inconsistent/broken gamma support.[57]
- 8.0 and below don't have color-correction support.[57]

Operating system support for PNG icons

PNG icons have been supported in most distributions of Linux since at least 1999, in desktop environments such as GNOME.[69] In 2006, Microsoft Windows support for PNG icons was introduced in Windows Vista.[70] PNG icons are supported in AmigaOS 4, AROS, macOS, iOS and MorphOS as well. In addition, Android makes extensive use of PNGs.

File size and optimization software

PNG file size can vary significantly depending on how it is encoded and compressed; this is discussed and a number of tips are given in *PNG: The Definitive Guide*.[49]

Compared to GIF

Compared to GIF files, a PNG file with the same information (256 colors, no ancillary chunks/metadata), compressed by an effective compressor is normally smaller than a GIF image. Depending on the file and the compressor, PNG may range from somewhat smaller (10%) to significantly smaller (50%) to somewhat larger (5%), but is rarely significantly larger[49] for large images. This is attributed to the performance of PNG's DEFLATE compared to GIF's LZW, and because the added precompression layer of PNG's predictive filters take account of the 2-dimensional image structure to further compress files; as filtered data encodes differences between pixels, they will tend to cluster closer to 0, rather than being spread across all possible values, and thus be more easily compressed by DEFLATE. However, some versions of Adobe Photoshop, CorelDRAW and MS Paint provide poor PNG compression, creating the impression that GIF is more efficient.[49]

File size factors

PNG files vary in size due to a number of factors:

color depth
: Color depth can range from 1 to 64 bits per pixel.

ancillary chunks
: PNG supports metadata—this may be useful for editing, but unnecessary for viewing, as on websites.

interlacing
: As each pass of the Adam7 algorithm is separately filtered, this can increase file size.[49]

filter
: As a precompression stage, each line is filtered by a predictive filter, which can change from line to line. As the ultimate DEFLATE step operates on the whole image's filtered data, one cannot optimize this row-by-row; the choice of filter for each row is thus potentially very variable, though heuristics exist.[note 1]

compression
: With additional computation, DEFLATE compressors can produce smaller files.

There is thus a filesize trade-off between high color depth, maximal metadata (including color space information, together with information that does not affect display), interlacing, and speed of compression, which all yield large files, with lower color depth, fewer or no ancillary chunks, no interlacing, and tuned but computationally intensive filtering and compression. For different purposes, different trade-offs are chosen: a maximal file may be best for archiving and editing, while a stripped down file may be best for use on a website, and similarly fast but poor compression is preferred when repeatedly editing and saving a file,

while slow but high compression is preferred when a file is stable: when archiving or posting. Interlacing is a trade-off: it dramatically speeds up early rendering of large files (improves latency), but may increase file size (decrease throughput) for little gain, particularly for small files.[49]

Lossy PNG compression

Although PNG is a lossless format, PNG encoders can preprocess image data in a lossy fashion to improve PNG compression. For example, quantizing a truecolor PNG to 256 colors allows the indexed color type to be used for a likely reduction in file size.[71]

Image editing software

Some programs are more efficient than others when saving PNG files, this relates to implementation of the PNG compression used by the program.

Many graphics programs (such as Apple's Preview software) save PNGs with large amounts of metadata and color-correction data that are generally unnecessary for Web viewing. Unoptimized PNG files from Adobe Fireworks are also notorious for this since they contain options to make the image editable in supported editors. Also CorelDRAW (at least version 11) sometimes produces PNGs which cannot be opened by Internet Explorer (versions 6–8).

Adobe Photoshop's performance on PNG files has improved in the CS Suite when using the Save For Web feature (which also allows explicit PNG/8 use).

Adobe's Fireworks saves larger PNG files than many programs by default. This stems from the mechanics of its *Save* format: the images produced by Fireworks' save function include large, private chunks, containing complete layer and vector information. This allows further lossless editing. When saved with the *Export* option, Fireworks' PNGs are competitive with those produced by other image editors, but are no longer editable as anything but flattened bitmaps. Fireworks is unable to save size-optimized vector-editable PNGs.

Other notable examples of poor PNG compressors include:

- Microsoft's Paint for Windows XP
- Microsoft Picture It! Photo Premium 9

Poor compression increases the PNG file size but does not affect the image quality or compatibility of the file with other programs.

When the color depth of a truecolor image is reduced to an 8-bit palette (as in GIF), the resulting image data is typically much smaller. Thus a truecolor PNG is typically larger than a color-reduced GIF, although PNG could store the color-reduced version as a palettized file of comparable size. Conversely, some tools, when saving images as PNGs, automatically save them as truecolor, even if the original data use only 8-bit color, thus bloating the file unnecessarily.[49] Both factors can lead to the misconception that PNG files are larger than equivalent GIF files.

Optimizing tools

Various tools are available for optimizing PNG files; they do this by:

- (optionally) removing ancillary chunks,
- reducing color depth, either:
  - use a palette (instead of RGB) if the image has 256 or fewer colors,
  - use a smaller palette, if the image has 2, 4, or 16 colors, or
  - (optionally) lossily discard some of the data in the original image,
- optimizing line-by-line filter choice, and
- optimizing DEFLATE compression.

Tool list

- pngcrush is the oldest of the popular PNG optimizers. It allows for multiple trials on filter selection and compression arguments, and finally chooses the smallest one. This working model is used in almost every png optimizer.
- advpng and the similar advdef utility in the AdvanceCOMP package recompress the PNG IDAT. Different DEFLATE implementations are applied depending on the selected compression level, trading between speed and file size: zlib at level 1, libdeflate at level 2, 7-zip's LZMA DEFLATE at level 3, and zopfli at level 4.
- pngout was made with the author's own deflater (same to the author's zip utility, kzip), while keeping all facilities of color reduction / filtering. However, pngout doesn't allow for using several trials on filters in a single run. It's suggested to use its commercial GUI version, pngoutwin, or used with a wrapper to automate the trials or to recompress using its own deflater while keep the filter line by line.[note 2]
- zopflipng was also made with a self-own deflater, zopfli. It has all the optimizing features pngcrush has (including automating trials) while providing a very good, but slow deflater.

A simple comparison of their features is listed below.

Optimizer	Chunk removal	Color reduction	Filtering	Filter reuse[note 3]	Multiple trials on filters in a single run	Deflater[note 4]
advpng	Yes	No[note 5]	0	No	N/A[note 6]	(multiple)
advdef	No	No	Reuses previous filter set	Always	N/A	(multiple)
pngcrush	Yes	Yes	0–4 or adaptive	No	Yes	zlib
pngout	Yes	Yes	0–4 or adaptive	Yes[note 2]	No	kzip
zopflipng	Yes	Yes	0–4 or adaptive with 2 different algorithms, or with a brute way	Yes	Yes	zopfli

Before zopflipng was available, a good way in practice to perform a png optimization is to use a combination of 2 tools in sequence for optimal compression: one which optimizes filters (and removes ancillary chunks), and one which optimizes DEFLATE. Although pngout offers both, only one type of filter can be specified in a single run, therefore it can be used with a wrapper tool or in combination with pngcrush,[note 2] acting as a re-deflater, like advdef.

Ancillary chunk removal

For removing ancillary chunks, most PNG optimization tools have the ability to remove all color correction data from PNG files (gamma, white balance, ICC color profile, standard RGB color profile). This often results in much smaller file sizes. For example, the following command line options achieve this with pngcrush:

`pngcrush -rem gAMA -rem cHRM -rem iCCP -rem sRGB` *`InputFile.png OutputFile.png`*

Filter optimization

pngcrush, pngout, and zopflipng all offer options applying one of the filter types 0–4 globally (using the same filter type for all lines) or with a "pseudo filter" (numbered 5), which for each line chooses one of the filter types 0–4 using an adaptive algorithm. Zopflipng offers 3 different adaptive method, including a brute-force search that attempts to optimize the filtering.[note 7]

pngout and zopflipng provide an option to preserve/reuse[note 2][note 8] the line-by-line filter set present in the input image.

pngcrush and zopflipng provide options to try different filter strategies in a single run and choose the best. The freeware command line version of pngout doesn't offer this, but the commercial version, pngoutwin, does.[note 9]

DEFLATE optimization

Zopfli and the LZMA SDK provide DEFLATE implementations that can produce higher compression ratios than the zlib reference implementation at the cost of performance. AdvanceCOMP's `advpng` and `advdef` can use either of these libraries to re-compress PNG files. Additionally, PNGOUT contains its own proprietary DEFLATE implementation.

`advpng` doesn't have an option to apply filters and always uses filter 0 globally (leaving the image data unfiltered); therefore it should not be used where the image benefits significantly from filtering. By contrast, `advdef` from the same package doesn't deal with PNG structure and acts only as a re-deflater, retaining any existing filter settings.

Icon optimization

Since icons intended for Windows Vista and later versions may contain PNG subimages, the optimizations can be applied to them as well. At least one icon editor, Pixelformer, is able to perform a special optimization pass while saving ICO files, thereby reducing their sizes. FileOptimizer (mentioned above) can also handle ICO files.

Icons for macOS may also contain PNG subimages, yet there isn't such tool available.

See also

- Computer graphics, including:
- Image editing
- Image file formats
- Related graphics file formats

    - APNG Animated PNG
    - JPEG Network Graphics (JNG)
    - Multiple-image Network Graphics (MNG)
- Similar file formats
    - X PixMap for portable icons
- Scalable Vector Graphics
- WebP

Explanatory notes

1. The filtering is used to increase the similarity to the data, hence increasing the compression ratio. However, there is theoretically no formula for similarity, nor absolute relationship between the similarity and compressor, thus unless the compression is done, one can't tell one filter set is better than another.
2. Use pngout -f6 to reuse previous filter set
3. The tools offering such feature could act as a pure re-deflater to PNG files.
4. zlib, the reference deflate implementation, compression is suboptimal even at the maximum level. See Zopfli, zip format in 7-zip and pngout.
5. Not only does advpng not support color reduction, it also fails on images with a reduced colorspace.
6. Advpng can only apply filter 0 globally, thus it's neither yes or no, but N/A.
7. [pngcrush|pngout] -f OR zopflipng --filters
8. zopflipng --filters=p
9. Pngoutwin's setting dialog for optimization offers the user a selection of filter strategies.

References

1. "ISO/IEC 15948:2004 – Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification" (http://www.iso.org/iso/catalogue_detail.htm?csnumber=29581). International Organization for Standardization. 3 March 2004. Retrieved 19 February 2011.
2. Roelofs, Greg (29 May 2010). "History of PNG" (http://www.libpng.org/pub/png/#history). libpng. Retrieved 20 October 2010.
3. W3C 2003, 1 Scope (https://www.w3.org/TR/PNG/#1Scope).
4. "Definition of PNG noun from the Oxford Advanced Learner's Dictionary" (https://www.oxfordlearnersdictionaries.com/definition/english/png). *Oxford Learner's Dictionaries*. Retrieved 21 January 2018.
5. "Portable Network Graphic .PNG File Description" (https://surferhelp.goldensoftware.com/subsys/subsys_portable_network_graphic_file_description.htm). *surferhelp.goldensoftware.com*. Retrieved 12 August 2022.
6. Roelofs, Greg. "Web Review: PNG's NOT GIF!" (https://people.apache.org/~jim/NewArchitect/webrevu/1997/05_09/designers/05_09_97_1.html). *people.apache.org*. Retrieved 24 November 2021.
7. T. Boutell; et al. (March 1997). "PNG (Portable Network Graphics) Specification Version 1.0" (https://tools.ietf.org/html/rfc2083#section-3). *RFC 2083*. IESG. sec. 3. doi:10.17487/RFC2083 (https://doi.org/10.17487%2FRFC2083).

8. "Registration of new Media Type image/png" (https://www.iana.org/assignments/media-types/image/png). IANA. 27 July 1996.
9. Roelofs 1999, Chapter 7. History of the Portable Network Graphics Format.
10. W3C 2003, 11.2.2 IHDR Image header (https://www.w3.org/TR/PNG/#11IHDR)
11. Roelofs, Greg (29 September 2011). "Portable Network Graphics (PNG) Specification and Extensions" (http://www.libpng.org/pub/png/spec/). *libpng*. Retrieved 15 August 2021.
12. "PNG (Portable Network Graphics) Specification 1.0" (https://tools.ietf.org/html/rfc2083#page-39). W3C. 1 October 1996. sec. 8.4. "PNG itself is strictly a single-image format. (...) In the future, a multiple-image format based on PNG may be defined. Such a format will be considered a separate file format"
13. Boutell, Thomas (1 October 1996). "PNG (Portable Network Graphics) Specification 1.0" (http://tools.ietf.org/html/rfc2083).
14. W3C 2003, 5.2 PNG signature (https://www.w3.org/TR/PNG/#5PNG-file-signature).
15. W3C 2003, 5.3 Chunk layout (https://www.w3.org/TR/PNG/#5Chunk-layout).
16. Laphroaig, Manul (31 October 2017). *PoC or GTFO* (https://books.google.com/books?id=lvMxDwAAQBAJ&pg=PT686). ISBN 9781593278984. "Each chunk consists of four parts: Length, a Chunk Type, the Chunk Data, and a 32-bit CRC. The Length is a 32-bit unsigned integer indicating the size of only the Chunk Data field"
17. Laphroaig, Manul (31 October 2017). - *PoC or GTFO* (https://books.google.com/books?id=lvMxDwAAQBAJ&pg=PT686). ISBN 9781593278984. "Chunk Type is a 32-bit FourCC code such as IHDR, IDAT, or IEND." {{cite book}}: Check |url= value (help)
18. W3C 2003, 11.2.4 IDAT Image data (https://www.w3.org/TR/PNG/#11IDAT).
19. W3C 2003, 11.2.5 IEND Image trailer (https://www.w3.org/TR/PNG/#11IEND).
20. W3C 2003, 11.3.3.3 iCCP Embedded ICC profile (https://www.w3.org/TR/PNG/#11iCCP).
21. Thomas Kopp (17 April 2008). "PNG Digital Signatures: Extension Specification" (http://libpng.download/documents/signatures/).
22. "Extensions to the PNG 1.2 Specification, version 1.5.0" (http://ftp-osl.osuosl.org/pub/libpng/documents/pngext-1.5.0.html). *ftp-osl.osuosl.org*.
23. W3C 2003, 11.3.3.2 gAMA Image gamma (https://www.w3.org/TR/PNG/#11gAMA).
24. W3C 2003, 11.3.5.3 pHYs Physical pixel dimensions (https://www.w3.org/TR/PNG/#11pHYs).
25. W3C 2003, 11.3.3.4 sBIT Significant bits (https://www.w3.org/TR/PNG/#11sBIT).
26. "PNG (Portable Network Graphics) Specification \ Version 1.0" (https://www.w3.org/TR/PNG-Chunks.html). *w3.org*. Retrieved 30 May 2022. 4.2.6. sBIT Significant bits, 13 bytes total - color type 2 and 3 totaled 6 bytes
27. Roelofs 2003, Significant Bits (sBIT) (http://www.libpng.org/pub/png/book/chapter11.html#png.ch11.div.7)"Grayscale images are the simplest; sBIT then contains a single byte indicating the number of significant bits in the source data"
28. "PNG Specification: Chunk Specifications" (http://www.libpng.org/pub/png/spec/1.2/PNG-Chunks.html#C.sRGB).
29. "PNG News from 2006" (http://www.libpng.org/pub/png/png2006.html). Libpng.org.
30. "PNG Specification: Rationale" (http://www.w3.org/TR/PNG-Rationale.html). *w3.org*.
31. W3C 2003, 9 Filtering (https://www.w3.org/TR/PNG/#9Filters).
32. "Filter Algorithms" (http://www.libpng.org/pub/png/spec/1.2/PNG-Filters.html). *PNG Specification*.

33. Paeth, Alan W. (1991). Arvo, James (ed.). "Image File Compression Made Easy". *Graphics Gems 2*. Academic Press, San Diego: 93–100. doi:10.1016/B978-0-08-050754-5.50029-3 (https://doi.org/10.1016%2FB978-0-08-050754-5.50029-3). ISBN 0-12-064480-0.
34. Crocker, Lee Daniel (July 1995). "PNG: The Portable Network Graphic Format" (http://www.ddj.com/architect/184409587?pgno=4). *Dr. Dobb's Journal*. **20** (232): 36–44.
35. "Introduction to PNG" (http://nuwen.net/png.html). nuwen.net. Retrieved 20 October 2010.
36. "Can I use... Support tables for HTML5, CSS3, etc" (https://caniuse.com/apng). *caniuse.com*. Retrieved 6 February 2021.
37. "iOS 8 and iPhone 6 for web developers and designers: next evolution for Safari and native webapps" (http://www.mobilexweb.com/blog/safari-ios8-iphone6-web-developers-designers). mobilexweb.com. 17 September 2014. Retrieved 24 September 2014.
38. scroggo (14 March 2017). "chromium / chromium / src / 7d2b8c45afc9c0230410011293cc2e1dbb8943a7" (https://chromium.googlesource.com/chromium/src/+/7d2b8c45afc9c0230410011293cc2e1dbb8943a7). *chromium.googlesource.com*. Retrieved 31 March 2017.
39. chrome-cron; et al. (27 March 2017). "chromium / chromium / src / 59.0.3047.0..59.0.3053.0" (https://chromium.googlesource.com/chromium/src/+log/59.0.3047.0..59.0.3053.0?pretty=fuller&n=10000). *chromium.googlesource.com*. Retrieved 31 March 2017.
40. "Dev.Opera — What's new in Chromium 59 and Opera 46" (https://dev.opera.com/blog/opera-46/). *dev.opera.com*. Retrieved 11 September 2022.
41. "Vote failed: APNG 20070405a" (https://web.archive.org/web/20080203042347/http://sourceforge.net/mailarchive/message.php?msg_name=3.0.6.32.20070420132821.012dd8e8%40mail.comcast.net). 20 April 2007. Archived from the original (http://sourceforge.net/mailarchive/message.php?msg_name=3.0.6.32.20070420132821.012dd8e8%40mail.comcast.net) on 3 February 2008.
42. "PNG Group animation proposal comparison + test-software" (https://web.archive.org/web/20090124013928/http://gjuyn.xs4all.nl/pnganim.html). *xs4all.nl*. Archived from the original (http://gjuyn.xs4all.nl/pnganim.html) on 24 January 2009.
43. Deutsch, L. Peter; Gailly, Jean-Loup (May 1996). "rfc1950" (https://datatracker.ietf.org/doc/html/rfc1950#page-4). *datatracker.ietf.org*. Retrieved 18 August 2021.
44. Deutsch, L. Peter (May 1996). "rfc1951" (https://datatracker.ietf.org/doc/html/rfc1951#page-3). *datatracker.ietf.org*. Retrieved 18 August 2021.
45. "A Basic Introduction to PNG Features" (http://www.libpng.org/pub/png/pngintro.html). Libpng.org. Retrieved 20 October 2010.
46. "GIF, PNG, JPG. Which One To Use?" (http://www.sitepoint.com/gif-png-jpg-which-one-to-use/). Sitepoint.com. 3 August 2009. Retrieved 20 October 2010.
47. "Extensions to the PNG 1.2 Specification, Version 1.5.0" (http://ftp-osl.osuosl.org/pub/libpng/documents/pngext-1.5.0.html#C.eXIf). Retrieved 5 May 2020.
48. "T.87 : Lossless and near-lossless compression of continuous-tone still images – Baseline" (http://www.itu.int/net/ITU-T/sigdb/speimage/T87.htm). International Telecommunication Union. Retrieved 20 March 2011.
49. Roelofs 2003, Chapter 9. Compression and Filtering (http://www.libpng.org/pub/png/book/chapter09.html)
50. "libpng" (http://www.libpng.org/pub/png/libpng.html). Retrieved 13 July 2013.
51. "Use of PNG Images to Display Data" (http://oregon.usgs.gov/png_images.html). Oregon Water Science Center. 16 February 2006.
52. "Why There Are No GIF files on GNU Web Pages" (https://www.gnu.org/philosophy/gif.html). *GNU Operating System*. 16 December 2008.

53. "PNG Fact Sheet" (http://www.w3.org/Press/PNG-fact.html). World Wide Web Consortium. 7 October 1996.
54. "Resource page for gif2png 2.5.11" (http://www.catb.org/~esr/gif2png/). *catb.org*.
55. "Burn All GIFs" (http://burnallgifs.org/archives/). *burnallgifs.org*.
56. "PNG Transparency in Internet Explorer" (https://www.pcmag.com/article2/0,2817,1645187,00.asp). *PC Magazine*. 5 October 2004.
57. "Browsers with PNG Support" (http://www.libpng.org/pub/png/pngapbr.html#msie-win-unix). 14 March 2009.
58. "Windows Explorer Crashes When I Click on a Fireworks PNG File to View It" (http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_13501&sliceId=2). Adobe Systems. 5 June 2007.
59. "Unable to view .png images with Internet Explorer 4.0" (http://support.microsoft.com/kb/174946). *Microsoft Knowledge Base*.
60. "PNGs That Are Inside of an Object Tag Print as a Negative Image" (http://support.microsoft.com/kb/257081). *Microsoft Knowledge Base*.
61. "PNG Images Are Printed Improperly in Internet Explorer 5.01" (http://support.microsoft.com/kb/255239). *Microsoft Knowledge Base*.
62. "You cannot view some PNG images in Internet Explorer 6" (http://support.microsoft.com/kb/822071). *Microsoft Knowledge Base*.
63. "You cannot use Internet Explorer 6 to open a PNG file that contains one or more zero-length IDAT chunks" (http://support.microsoft.com/kb/897242). *Microsoft Knowledge Base*.
64. "PNG Frequently Asked Questions" (http://www.libpng.org/pub/png/pngfaq.html#msie).
65. "PhD: Portable Network Graphics Lose Transparency in Web Browser" (http://support.microsoft.com/kb/265221). *Microsoft Knowledge Base*.
66. "PNG Files Do Not Show Transparency in Internet Explorer" (http://support.microsoft.com/kb/294714). *Microsoft Knowledge Base*.
67. Lovitt, Michael (21 December 2002). "Cross-Browser Variable Opacity with PNG: A Real Solution" (https://web.archive.org/web/20110818032515/http://www.alistapart.com/articles/pngopacity/). *A List Apart*. Archived from the original (http://www.alistapart.com/articles/pngopacity/) on 18 August 2011. Retrieved 21 July 2009.
68. "IE7 alpha transparent PNG + opacity" (https://web.archive.org/web/20110827114121/http://channel9.msdn.com/forums/TechOff/257324-IE7-alpha-transparent-PNG--opacity). *Channel 9*. Archived from the original (http://channel9.msdn.com/forums/TechOff/257324-IE7-alpha-transparent-PNG--opacity/) on 27 August 2011. Retrieved 23 January 2009.
69. Fulbright, Michael (1999). "GNOME 1.0 Library Roadmap" (https://web.archive.org/web/20100130042852/http://developer.gnome.org/doc/whitepapers/libroadmap/). Archived from the original (http://developer.gnome.org/doc/whitepapers/libroadmap/) on 30 January 2010. Retrieved 19 December 2007.
70. "Windows Vista – Icons" (https://web.archive.org/web/20071111153449/http://www.oone.googlepages.com/windows_vista_icons.htm). *OOne*. 2007. Archived from the original (http://www.oone.googlepages.com/windows_vista_icons.htm) on 11 November 2007. Retrieved 12 November 2007.
71. "PNG can be a lossy format" (http://pngmini.com/lossypng.html). Pngmini.com. Retrieved 1 February 2014.

Further reading

- Roelofs, Greg (April 1997). "Linux Gazette: History of the Portable Network Graphics (PNG) Format" (http://linuxgazette.net/issue13/png.html). *Linux Journal*. Specialized Systems

Consultants, Inc. **1997** (36es). ISSN 1075-3583 (https://www.worldcat.org/issn/1075-3583).
- Roelofs, Greg (2003). *PNG: The Definitive Guide* (https://archive.org/details/pngdefinitivegui00roel) (2nd ed.). O'Reilly Media. ISBN 1-56592-542-4.
- "Portable Network Graphics (PNG) Specification" (http://www.w3.org/TR/PNG/) (Second ed.). W3C. 10 November 2003.

External links

- PNG Home Site (http://www.libpng.org/pub/png/)
- libpng Home Page (http://www.libpng.org/pub/png/libpng.html)
- *The Story of PNG* by Greg Roelofs (http://www.libpng.org/pub/png/slashpng-1999.html)
- Test inline PNG images (http://www.w3.org/Graphics/PNG/Inline-img.html)
- RFC 2083 (https://datatracker.ietf.org/doc/html/rfc2083)
- More information about PNG color correction (https://hsivonen.iki.fi/png-gamma/)
- The GD-library to generate dynamic PNG-files with PHP (http://php.net/gd)
- PNG Adam7 interlacing (http://schaik.com/png/adam7.html)
- Encoding Web Shells in PNG files (https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/): Encoding human readable data inside an IDAT block.