MNG-Lite (Multiple-image Network Graphics - Lite) Format Version 0.91

For list of authors, see Credits (Chapter 16).

Status of this Memo

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

It is a **proposal**, and the format is subject to change.

Comments on this document can be sent to the PNG specification maintainers at one of the following addresses:

- mpng-list@ccrc.wustl.edu
- png-group@w3.org
- png-info@uunet.uu.net

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/mng/documents/.`

Changes from fifty-sixth MNG draft (mng-0.91-19980202)

- Reformatted to allow extracting a "MNG-Lite" format, which is a proper subset of MNG that only allows "simple" MNG datastreams.

Abstract

This document presents a reduced version (MNG-Lite) of the format of a MNG (Multiple-image Network Graphics) datastream. MNG is a multiple-image member of the PNG (Portable Network Graphics) format family, that can contain animations (slide shows) comprised of PNG single-image datastreams.

A MNG frame normally contains a two-dimensional image or a two-dimensional layout of smaller images. It could also contain three-dimensional "voxel" data arranged as a series of two-dimensional planes (or tomographic slices), each plane being represented by a PNG datastream.

The MNG and JNG formats use the same chunk structure that is defined in the PNG specification, and they share other features of the PNG format. Any valid PNG or JNG datastream is also a valid MNG datastream.

This document includes examples that demonstrate various capabilities of MNG-Lite including simple movies and composite frames.

Table of Contents

1 Introduction **5**

2 Terminology **7**

3 MNG Chunks **9**
3.1 Critical MNG control chunks . . . 9
3.1.1 MHDR MNG datastream header . . . 9
3.1.2 MEND End of MNG datastream . . . 11
3.1.3 LOOP, ENDL Define a loop . . . 11
3.2 Critical MNG image defining chunks . . . 11
3.2.1 DEFI Define an object . . . 11
3.2.2 IHDR, PNG chunks, IEND . . . 13
3.2.3 JHDR, JNG chunks, IEND . . . 13
3.3 Critical MNG image displaying chunks . . . 14
3.3.1 BACK Background . . . 14
3.3.2 FRAM Frame definitions . . . 15
3.3.3 TERM Termination action . . . 22
3.4 SAVE and SEEK chunks . . . 23
3.5 Ancillary MNG chunks . . . 23
3.5.1 pHYs Physical pixel size . . . 23
3.5.2 PNG ancillary chunks . . . 24

4 The JPEG Network Graphics (JNG) Format **24**
4.1 JNG critical chunks . . . 24
4.1.1 JHDR JNG header . . . 25
4.1.2 JDAT JNG image data . . . 26
4.1.3 IDAT JNG alpha data . . . 29
4.1.4 JSEP 8-bit/12-bit image separator . . . 29
4.2 JNG ancillary chunks . . . 29

5 Chunk Copying Rules **30**

6 Minimum Requirements for MNG-Compliant Viewers **31**
6.1 Required PNG chunk support . . . 31
6.2 Required JNG chunk support . . . 32
6.3 Required MNG chunk support . . . 32

7 Recommendations for Encoders **33**
7.1 Use a common color space . . . 33
7.2 Use the right framing mode . . . 33
7.3 Including optional index in SAVE chunk . . . 33
7.4 Interleaving JDAT and IDAT chunks . . . 34

8 Recommendations for Decoders **34**
8.1 Note on compositing . . . 34
8.2 Decoder handling of fatal errors . . . 35
8.3 Decoder handling of interlaced images . . . 35
8.4 Decoder handling of palettes . . . 35
8.5 Behavior of single-frame viewers . . . 36
8.6 Clipping . . . 36

9 Recommendations for Editors **37**
9.1 Editing datastreams with optional index . . . 37

10 Miscellaneous Topics **37**
10.1 File name extension . . . 37
10.2 Internet media type . . . 37
10.3 Uniform Resource Identifier (URI) . . . 37

11 References **39**

12 Security Considerations **40**

13 Appendix: Examples **41**
13.1 Example 1: A single image . . . 41
13.2 Example 2: Very simple movie . . . 41
13.3 Example 3: Simple slideshow . . . 42
13.4 Example 4: A more storage-efficient slideshow . . . 42

14 Credits **43**

1 Introduction

This specification defines the format of the MNG (Multiple-image Network Graphics) and JNG (JPEG Network Graphics) datastreams.

Note: This specification depends on the PNG (Portable Network Graphics) [PNG]. and the JPEG (Joint Photographic Experts Group) specifications. The PNG specification is available at the PNG home page,

```
http://www.cdrom.com/pub/png/
```

A MNG datastream describes a sequence of single frames, each of which can be composed of zero or more embedded images or directives to show previously defined images.

A typical MNG datastream consists of:

- The 8-byte MNG signature.
- The MHDR chunk.
- Frame definitions. A frame is one or more subframes, the last of which has a nonzero interframe delay.
- Subframe definitions. A subframe contains one or more layers, together with clipping information and an interframe delay.
- Layer definitions. A layer is:
  - An embedded image.
  - A rectangle filled with the background color.
  - Sometimes, an empty rectangle.
- The MEND chunk.

MNG is pronounced "Ming."

When a MNG datastream is stored in a file, it is recommended that ".mng" be used as the file suffix. In network applications, the Media Type "video/x-mng" can be used. Registration of the media type "video/mng" might be pursued at some future date.

The first eight bytes of a MNG datastream are

```
138 77 78 71 13 10 26 10
```

(decimal) which is similar to the PNG signature with "\212 M N G" instead of "\211 P N G" in bytes 0–3.

MNG does not yet accommodate sound or complex sequencing information, but these capabilities might be added at a later date, in a backward-compatible manner. These issues are being discussed in the mpng-list@ccrc.wustl.edu mailing list.

Chunk structure (length, name, data, CRC) and the chunk-naming system are identical to those defined in the PNG specification. As in PNG, all integers that require more than one byte must be in network byte order.

The chunk copying rules for MNG employ the same mechanism as PNG, but with rules that are explained more fully (see below, Chapter 7). A MNG editor is not permitted to move unknown chunks across the SAVE and SEEK chunks, across any chunks that can cause images to be displayed, or into or out of a IHDR-IEND or similar sequence.

Note that decoders are not required to follow any decoding models described in this specification nor to follow the instructions in this specification, as long as they produce results identical to those that could be produced by a decoder that did use this model and did follow the instructions.

Each chunk of the MNG datastream or of any embedded object is an independent entity, i.e., no chunk is ever enclosed in the data segment of another chunk.

An independent PNG or JNG datastream, with a PNG or JNG signature, is also a valid MNG datastream that must be recognized and decoded by MNG-compliant decoders. This kind of MNG datastream will contain only a single embedded image.

Because the embedded objects making up a MNG are normally in PNG format, MNG shares the good features of PNG:

- It is unencumbered by patents.
- It is streamable.
- It has excellent, lossless compression.
- It stores up to four channels (red, green, blue, alpha), with up to 16 bits per channel.
- It provides transparency and an alpha channel.
- It provides platform-independent rendition of colors by inclusion of gamma and chromaticity information.
- It provides early detection of common file transmission errors and robust detection of file corruption.
- Single-image GIF files can be losslessly converted to PNG.
- It is complementary to JPEG and does not attempt to replace JPEG for lossy storage of images (however, MNG can accommodate JPEG-encoded images that are wrapped in the PNG-like JNG format that is defined herein).

In addition:

- It provides animation with variable interframe delays.
- It allows composition of frames containing multiple images.
- JNG provides JPEG with transparency, alpha, and color space information.

- Multiple-image GIF files (except for those using the "restore-to-previous" disposal method) can be losslessly converted to MNG.
- Most JPEG files can be losslessly converted to JNG or MNG, and all JNG datastreams can be losslessly converted to JPEG files.
- It is complementary to MPEG and does not attempt to replace MPEG for lossy storage of video (MNG does, however, provide the capability of including animations consisting of JPEG-encoded images that have been wrapped in the PNG format).

2 Terminology

See also the glossary in the PNG specification.

requirement levels
: The words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119, and the word "CAN" is equivalent to the word "MAY" as described therein. "NOT ALLOWED" and "NOT PERMITTED" describe conditions that "MUST NOT" occur. "ALLOWED" and "PERMITTED" describe conditions that "CAN" occur.

embedded object or image
: A concrete object or image that appears in-line in a MNG datastream.

frame
: A layout of a background and zero or more images, followed by a specified nonzero delay time or by the MEND chunk, that is to be displayed as a still frame or as part of an animation. An animation would ideally appear to a perfect observer (with an inhumanly fast visual system) as a sequence of still frames. A group of MNG images whose interframe delay time is zero makes up a "subframe" (see below) instead of a "frame". The final subframe in a segment completes a frame, regardless of its interframe delay.

frame area
: The portion of the display surface whose pixels are inside the subframe clipping boundaries as defined by the FRAM chunk (see the FRAM chunk specification below, Paragraph 4.3.2).

frame duration
: The amount of time a frame should be visible when an animation is played. In reality, it takes a nonzero amount of time to display a frame. No matter which moment is picked as the "start" of the frame, the frame duration measures the time to the "start" of the next frame.

frame origin
: The upper left corner of the output device (frame buffer, screen, window, page, etc.) where the pixels are to be displayed. This is the $\{0,0\}$ position for the purpose of defining frame clipping boundaries, image locations, and image clipping boundaries. Note that in a windowing system, the

frame origin might be moved offscreen, but the locations in DEFI chunks would still be measured from this offscreen origin.

image N or object N
Shorthand for "the object with the object attribute set pointed to by `'object_id=N'`". In MNG-Lite, only image 0 is permitted.

layer
A visible embedded image or a stored image that is displayed in response to a SHOW chunk directive, located with respect to the frame boundaries and clipped with respect to the subframe clipping boundaries and the image's own clipping boundaries, or the background image, clipped, located, and displayed against the background color at the beginning of a subframe. Note that a layer can be completely empty if the image is entirely outside the clipping boundaries. A layer can be thought of as a transparent rectangle with the same dimensions as the frame, with an image composited into it, or it can be thought of as a rectangle having the same dimensions (possibly zero) and location as those of the object after it has been located and clipped.

An embedded visible PNG or JNG datastream generates a single layer, even though it might be interlaced or progressive. If the background consists of both a background color and a background image, these are combined into a single layer.

object, object_id
An image or a nonviewable basis object. The `object_id` is an unsigned sixteen-bit number that serves as the identifier of a set of object attributes. In MNG-Lite, only image 0 is permitted.

potentially visible image
A potentially visible image is

- a not-yet-defined viewable object that is "marked" for on-the-fly display while the embedded image that defines it is being decoded, by setting its `do_not_show` flag to zero.

signal
An entity with a number that can arrive asynchronously at the decoder. More detailed semantics, like whether multiple signals of the same number (or even different numbers) can be queued, are beyond the scope of this specification.

subframe
A group of one or more layers, with nonzero delay time between them and following them, that are to be displayed as a part of a still frame or as part of an animation. Subframes are defined in MNG for convenience in applying frame parameters only to a subset of the layers making up a complete frame.

visible image
Actually drawn on a display. If an object is visible, a person looking at the display can see it.

3 MNG Chunks

This chapter describes chunks that can appear at the top level of a MNG datastream. Unless otherwise specified in the Delta-PNG chapter of this specification, they need not be recognized there.

3.1 Critical MNG control chunks

This section describes critical MNG control chunks. MNG-compliant decoders must recognize and process them.

3.1.1 MHDR MNG datastream header

The MHDR chunk is always first in all MNG datastreams except for those that consist of a PNG datastream with a PNG signature.

The MHDR chunk contains exactly 28 bytes:

```
Frame width:  4 bytes (unsigned integer).
Frame height: 4 bytes (unsigned integer).
Ticks per
 second:      4 bytes (unsigned integer).
Nominal layer
 count:       4 bytes (unsigned integer).
Nominal frame
 count:       4 bytes (unsigned integer).
Nominal play
 time:        4 bytes (unsigned integer).
Simplicity
 profile:     4 bytes:(unsigned integer).
                bit 0:
                       0: Presence or absence of any
                          features is unspecified.  All
                          other bits of the simplicity
                          profile must be zero (i.e, all
                          other even numbers are invalid).
                       1: Presence or absence of certain
                          features is specified by the
                          remaining bits of the simplicity
                          profile.
                bit 1:
                       0: complex MNG features are absent.
                bit 2:
                       0: transparency is absent or can be
                          ignored.
                       1: transparency is present and is
                          essential (critical) for proper
```

```
          display of the images.
bit 3:
       0: JNG is absent.
       1: JNG is present.
bit 4:
       0: Delta-PNG is absent.
bits 5-15:
       Reserved for public expansion.  Must
       be zero in this version.
bits 16-30:
       Available for private or experimental
       expansion.  Undefined in this version
       and can be ignored.
bit 31:
       Must be zero.
```

Decoders can ignore the "informative" frame-count, layer-count, play-time, and simplicity-profile fields.

The `frame_width` and `frame_height` fields give the intended display size (measured in pixels) and provide default clipping boundaries (see below, Paragraph 4.3.4). It is strongly recommended that these be set to zero if the MNG datastream contains no visible images.

The `ticks_per_second` field gives the unit used by the FRAM chunk to specify frame duration and sync timeout. It must be nonzero if the datastream contains an animation. When the datastream contains exactly one frame, this field should be set to zero. When this field is zero, the length of a tick is infinite, and decoders will ignore any attempt to define interframe delay, timeout, or any other variable that depends on the length of a tick. If the frames are intended to be displayed one at a time under user control, such as a slide show or a multi-page FAX, the tick length can be set to any positive number and a FRAM chunk can be used to set an infinite sync_timeout. Unless the user intervenes, viewers will only display the first frame in the datastream.

When `ticks_per_second` is nonzero, and there is no other information available about frame duration, viewers should display animations at the rate of one frame per tick.

If the frame-count field contains a zero, the frame count is unspecified. If it is nonzero, it contains the number of frames that would be displayed, ignoring the TERM chunk. If the frame count is greater than $2^{31} - 1$, encoders should write $2^{31} - 1$, representing an infinite frame count.

If the layer-count field contains a zero, the layer count is unspecified. If it is nonzero, it contains the number of layers in the datastream, ignoring the TERM chunk. If the layer count is greater than $2^{31} - 1$, encoders should write $2^{31} - 1$, representing an infinite layer count.

If the nominal-play-time field contains a zero, the nominal play time is unspecified. Otherwise, it gives the play time, in ticks, when the file is displayed ignoring the TERM chunk. Authors who write this field should choose a value of "ticks_per_second" that will allow the nominal play time to be expressed in a four-bit integer. If the nominal play time is greater than $2^{31} - 1$ ticks, encoders should write $2^{31} - 1$, representing an infinite nominal play time.

When the simplicity profile is zero, the simplicity (or complexity) of the MNG datastream is unspecified. (The simplicity profile must be nonzero in MNG-Lite files.)

If the simplicity profile is nonzero, it can be regarded as a 31-bit profile, with the ones bit being a "profile-validity" flag, the twos bit being a "simple MNG" flag, the fours bit being a "transparency" flag, the eights bit being a "JNG" flag, and the sixteens bit being a "Delta-PNG" flag. (Delta-PNG must not be present in MNG-Lite files.) The upper 15 bits (except for the most significant bit, which must be zero) are available for private test or experimental versions, and the remaining bits are reserved for future MNG versions, and must be zero in this version. If a bit is zero, the corresponding feature is guaranteed to be absent, and if a bit is one, the corresponding feature may be present in the MNG datastream.

"Transparency is absent or can be ignored" means that the MNG or PNG tRNS chunk is not present and no PNG or JNG image has an alpha channel, or that if they are present they are not essential (or critical) for displaying the images.

Encoders that write a nonzero simplicity profile should endeavor to be accurate, so that decoders that process it will not unnecessarily reject datastreams. For example, the simplicity profile 15 indicates that JNG, critical transparency, and at least one "complex" MNG feature are all present, but Delta-PNG is not. If the simplicity profile promises that certain features are absent, but they are actually present in the MNG datastream, the datastream is invalid.

3.1.2 MEND End of MNG datastream

The MEND chunk's data length is zero. It signifies the end of a MNG datastream.

3.1.3 LOOP, ENDL Define a loop

The LOOP chunk provides a "shorthand" notation that can be used to avoid having to repeat identical chunks in a MNG datastream. If the LOOP chunk is present, it can be ignored by MNG-Lite decoders, along with the ENDL chunk.

3.2 Critical MNG image defining chunks

3.2.1 DEFI Define an object

The DEFI chunk sets the default object attribute set (`object_id`, `potential_visibility`, `concrete_flag`, location, and clipping boundaries) for any subsequent images that are defined with IHDR-IEND, JHDR-IEND, or BASI-IEND datastreams. The DEFI chunk contains 2, 3, 4, 12, or 28 bytes:

```
Object id:      2 bytes (unsigned integer) identifier
                  to be given to the objects that
                  follow the DEFI chunk.
                  This field must be zero in MNG-Lite
                  files.

Do_not_show
```

```
 flag:          1 byte (unsigned integer)
                  0:  Make the objects potentially
                      visible.
                  1:  Do not make the objects potentially
                      visible.  This field can be omitted
                      if the concrete_flag, location, and
                      clipping boundary fields are also
                      omitted.  When it is omitted, the
                      image is potentially visible
                      (do_not_show=0).

Concrete flag: 1 byte (unsigned integer)
                  0:  Make the objects "abstract" (image
                      can not be the source for a
                      Delta-PNG)
                  1:  Make the objects "concrete" (object
                      can be the source for a Delta-PNG).
                  This field can be omitted if the
                  location and clipping boundary fields
                  are also omitted.  When it is omitted,
                  the object is made "abstract"
                  (concrete_flag=0).

X_location:    4 bytes (signed integer).
                  The X_location and Y_location fields can
                  be omitted if the clipping boundaries
                  are also omitted.  If so, decoders must
                  assume default values {X_location=0,
                  Y_location=0}.

Y_location:    4 bytes (signed integer).

Left_cb:       4 bytes (signed integer).  Left clipping
                  boundary.  The left_cb, right_cb,
                  top_cb, and bottom_cb fields can be
                  omitted as a group.  If so, decoders
                  must assume default values
                  {0, frame_width, 0, frame_height}.

Right_cb:      4 bytes (signed integer).

Top_cb:        4 bytes (signed integer).

Bottom_cb:     4 bytes (signed integer).
```

Negative values are permitted for the X and Y location and clipping boundaries. The positive directions are downward and rightward from the frame origin.

The DEFI chunk values are discarded after the object's IEND chunk is processed. If no DEFI chunk precedes an object, the decoder must use the following default values:

```
    Object_id = 0
  Do_not_show = 0
Concrete_flag = 0
   X_location = 0
   Y_location = 0
      Left_cb = 0
     Right_cb = frame_width
       Top_cb = 0
    Bottom_cb = frame_height
```

3.2.2 IHDR, PNG chunks, IEND

A PNG (Portable Network Graphics) datastream.

See the PNG specification [PNG] and the PNG Special Purpose Chunks document [PNG-EXT] for the format of the PNG chunks.

Any chunks between IHDR and IEND are written and decoded according to the PNG specification.

When the framing mode is 2 or 4 in the MNG FRAM chunk, the PNG oFFs and pHYs chunks and any future chunks that attempt to set the pixel dimensions or the drawing location must be ignored by MNG viewers and simply copied (according to the copying rules) by MNG editors. When the framing mode is 1 or 3 (i.e., when each PNG image is an individual frame), these chunks must be treated as described in the PNG specification.

The PNG gIFg, gIFt, and gIFx chunks must be ignored by viewers and must be copied according to the copying rules by MNG editors.

If `do_not_show=0` for the image when the IHDR chunk is encountered, a viewer can choose to display the image while it is being decoded, perhaps taking advantage of the PNG interlacing method, or to display it after decoding is complete.

3.2.3 JHDR, JNG chunks, IEND

A JNG (JPEG Network Graphics) datastream.

See the JNG specification below (Chapter 5) for the format of the JNG datastream.

Any chunks between JHDR and IEND are written and decoded according to the JNG specification.

The remaining discussion in the previous paragraph about PNG datastreams also applies to JNG datastreams.

3.3 Critical MNG image displaying chunks

3.3.1 BACK Background

The BACK chunk suggests a background color against which transparent, clipped, or less-than-full-frame images can be displayed.

```
Red_background:   2 bytes (unsigned integer).

Green_background: 2 bytes (unsigned integer).

Blue_background:  2 bytes (unsigned integer).

Mandatory
  background:     1 byte (unsigned integer).
                     0: Background color is advisory.
                        Applications can use it if they
                        choose to.
                     1: Background color is mandatory.
                        Applications must use it.
                     This byte can be omitted.  If so,
                     the background color is advisory.
```

Viewers are expected to composite every subframe in the MNG datastream against a fresh copy of the background, if the framing mode given in the FRAM chunk is 3 or 4. The images and the background are both clipped to the frame boundaries given in the FRAM chunk.

The three BACK components are always interpreted in the current color space as defined by any top-level gAMA, cHRM, iCCP, sRGB chunks that have appeared prior to the BACK chunk in the MNG datastream. If no such chunks appear, the color space is unknown.

When the BACK chunk appears between FRAM chunks, it applies to the upcoming frame, not to the current one. When the framing_mode is 3, it takes effect immediately prior to the next IHDR, JHDR, or BASI chunk in the datastream.

Multiple instances of the BACK chunk are permitted in a MNG datastream.

The BACK chunk can be omitted. If a background is required and the BACK chunk is omitted, then the viewer must supply its own background.

In practice, most applications that use MNG as part of a larger composition should ignore the BACK data if `mandatory_background=0` and the application already has its own background definition. This will frequently be the case in World Wide Web pages, to achieve nonrectangular transparent animations displayed against the background of the page.

3.3.2 FRAM Frame definitions

The FRAM chunk provides information that a decoder needs for generating layers, subframes, and frames. The FRAM parameters govern how the decoder is to behave when it encounters a FRAM chunk, a SHOW chunk, or an embedded image. The FRAM chunk also delimits subframes.

An empty FRAM chunk is just a subframe delimiter. A nonempty one is a subframe delimiter, and it also changes FRAM parameters, either for the upcoming subframe or until reset. When the FRAM chunk is not empty, it contains a framing-mode byte, an optional name string, a zero-byte separator, plus four 1-byte fields plus a variable number of optional fields.

```
Framing mode:  1 byte.

                0:  Don't change framing mode.

                1:  Each embedded image
                    generates a subframe consisting of
                    a single image layer.  There is no
                    background layer.  Use this mode to
                    avoid unnecessary clearing of the
                    display when the first image covers
                    the entire frame area, and
                    subsequent frames can be displayed
                    properly by simply overlaying them
                    on the prior frame.  This is the
                    default framing mode.

                2:  The group of embedded images
                    appearing prior
                    to the next FRAM chunk form a
                    composite subframe consisting of
                    zero or more image layers.  The
                    level, or stacking order, of each
                    layer is given by its order of
                    appearance in the datastream.
                    No interframe delay occurs between
                    the layers.

                3:  This is the same as framing_mode=1,
                    except that a background layer is
                    generated.  Each embedded image
                    generates a subframe
                    consisting of a background layer
                    followed by an image layer.  No
                    interframe delay occurs between the
                    background layer and the image layer.

                4:  This is the same as framing_mode=2,
```

```
                    except that a background layer is
                    generated prior to the image layers.

Subframe name: 0 or more bytes (Latin-1 Text).  Can be
                 omitted; if so, the subframe is nameless.

Separator:     1 byte:  (null).  Must be omitted if all
                 remaining fields are also omitted.

Change interframe
    delay:     1 byte.

                 0:  No.
                 1:  Yes, for the next subframe only.
                 2:  Yes, also reset default.

               This field and the next three must be
               omitted as a group if no frame parameters
               other than the framing mode are changed.

Change sync timeout and
  termination: 1 byte
                 0:  No.
                 1:  Deterministic, for the next subframe
                     only.
                 2:  Deterministic, also reset default.
                 3:  Decoder-discretion, for the next
                     subframe only.
                 4:  Decoder-discretion, also reset
                     default.
                 5:  User-discretion, for the next subframe
                     only.
                 6:  User-discretion, also reset default.
                 7:  External-signal, for the next subframe
                     only.
                 8:  External-signal, also reset default.

Change subframe clipping
 boundaries:   1 byte.
                 0:  No.
                 1:  Yes, for the next subframe only.
                 2:  Yes, also reset default.

Change sync id
  list:        1 byte.
                 0:  No.
                 1:  Yes, for this subframe only.
                 2:  Yes, also reset default list.
```

```
Interframe
  delay:        4 bytes (unsigned integer).  Must
                  be omitted if change_interframe_delay=0.
                  The range is [0..2^31-1] ticks.

Sync timeout:   4 bytes (unsigned integer).  Omit if
                  change_sync_timeout=0.  The range is
                  [0..2^31-1].  The value 2^31-1
                  (0x7fffffff) ticks represents an
                  infinite timeout period.

Subframe boundary
  delta type:   1 byte (unsigned integer).
                  0: Subframe clipping boundary values are
                     given directly.
                  1: Subframe clipping boundaries
                     are determined by adding the FRAM
                     data to their previous values.
                This and the following four
                fields must be omitted if
                change_frame_clipping_boundaries=0.

Left_fb or delta
  left_fb:      4 bytes (signed integer).

Right_fb or delta
  right_fb:     4 bytes (signed integer).

Top_fb or delta
  top_fb:       4 bytes (signed integer).

Bottom_fb or delta
  bottom_fb:    4 bytes (signed integer).

Sync id:        4 bytes (unsigned integer).  Omit if
                  change_sync_id_list=0 or if the new
                  list is empty; repeat until all
                  sync_id's have been listed.  The
                  range is [0..2^31-1].
```

When the FRAM parameters are changed, the new parameters affect the subframe that is about to be defined, not the one that is being terminated by the FRAM chunk.

Framing modes:

Framing mode 1.

When `framing_mode=1`, each image generates a separate subframe consisting of a single image layer. In the usual case, the interframe delay is nonzero, so each subframe is a frame. FRAM chunks need not appear to separate them.

The following events generate a subframe:

- Decoding a IHDR-IEND sequence at the MNG level, when it defines a potentially visible image.
- Decoding a JHDR-IEND sequence at the MNG level, when it defines a potentially visible image.
- Decoding a BASI-IEND sequence, when it defines a potentially visible image.

Any of these events generates a subframe, even if the visible image is outside the clipping boundaries and no pixels are actually changed.

For example (assuming that objects 1 through 5 are all viewable objects and a nonzero interframe delay time), the sequence

```
FRAM 1
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
```

will generate five frames, each containing a single subframe consisting of an image layer.

If the BACK chunk is present, encoders should insert a background layer, with a zero delay, ahead of the first image layer in a segment, even when the framing_mode is 1. Such layers must be included in the layer count but not in the frame count. If the BACK chunk is not present, the contents of the display become undefined at the beginning of each segment.

Framing mode 2.

Framing mode 2 is the same as framing mode 1, except that the interframe delay occurs between subframes rather than between images. In the usual case, the interframe delay is nonzero, so the subframes are composite frames. When `framing_mode=2`, viewers are expected to display all of the images at once, if possible, or as fast as can be managed, without clearing the display or restoring the background. The next FRAM chunk delimits the subframe. A subframe boundary also occurs when a SEEK chunk or the MEND chunk appears.

For example, the sequence

```
FRAM 2
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
IHDR ... IDAT ... IEND
FRAM
```

will result in a single subframe containing five layers, each consisting of one image displayed according to its location and CLIP data. If the images do not cover the entire frame, whatever was already on the display shows through.

When images in a subframe overlap, viewers are expected to composite the later images against the partially completed subframe that includes all earlier images.

This `framing_mode` is fundamentally declarative; it describes the elements that go into an individual subframe. It is up to the decoder to work out an efficient way of making the screen match the desired composition. Simple decoders can handle it as if it were procedural, compositing the images into the frame buffer in the order that they appear, but efficient decoders might do something different, as long as the final appearance of the frame is the same.

If the BACK chunk is present, encoders should insert a background layer, with a zero delay, ahead of the first image layer in a segment, even when the framing_mode is 2. Such layers must be included in the layer count but not in the frame count. If the BACK chunk is not present, the contents of the display become undefined at the beginning of each segment.

Framing mode 3.

When `framing_mode=3`, a subframe consisting of the background layer and an image layer is generated for each image. A subframe boundary occurs after each image appears. Otherwise, `framing_mode=3` is identical to `framing_mode=1`. Subframes are triggered by the same events that trigger a subframe when `framing_mode=1`.

You can use this mode to show a frame containing only the background, with its own time delay, as in

In this example, the background and the five images will be displayed one at a time against the background, like cards being dealt.

Framing mode 4.

When `framing_mode=4`, a subframe boundary and interframe delay occurs when each FRAM chunk appears. A background layer, consisting of the background image composited against the background color, is generated immediately after the FRAM chunk, Otherwise, `framing_mode=4` is identical to `framing_mode=2`. A subframe boundary also occurs when a SEEK chunk or the MEND chunk appears, but neither of these generates a background layer.

You can make a composite frame consisting of six layers (a background layer and 5 images) with

```
FRAM 4     (show background, no delay)
FRAM       (interframe delay, then start next frame)
```

Summary of framing modes.

This table summarizes the behavior of a viewer under the various framing modes.

Framing mode	Restore background	Interframe delay
1	Before first image* after first FRAM in the datastream	Before each FRAM after the first in the datastream

	(or segment, if jumped to segment)	
2	Before first image after first FRAM in the datastream (or segment, if jumped to segment)	Before each image after the first in the datastream
3	Before each image	Before each image after the first in the datastream
4	Before first image following each FRAM chunk	Before each FRAM after the first in the datastream

*"image" means an image layer that is generated in response to decoding an embedded object, even if no pixels are actually drawn due to the image being outside the clipping boundaries.

The framing_mode also affects the way decoders handle the pHYs chunk (see below, Paragraph 4.5.4).

The subframe name must conform to the same formatting rules as those for a PNG tEXt keyword: It must consist only of printable Latin-1 characters and must not have leading or trailing blanks, but can have single embedded blanks. There must be at least one (unless the subframe name is omitted) and no more than 79 characters in the keyword. Keywords are case-sensitive. There is no null byte within the keyword. Applications can use this field for such purposes as constructing an external list of subframes in the datastream. The subframe name only applies to the upcoming subframe; subsequent subframes are unnamed unless they also have their own frame_name field. It is recommended that the same name not appear in any other FRAM chunk. Subframe names should not begin with the case-insensitive strings "clock(", "frame(", or "frames(", which are reserved for use in URI queries and fragments.

The interframe delay value is the desired minimum time to elapse from the beginning of displaying one frame until the beginning of displaying the next frame. When the interframe delay is nonzero, which will probably be the usual case, subframes are frames. When it is zero, a frame consists of any number of consecutive subframes, until a nonzero delay subframe is encountered and completed. Decoders are not obligated to display such subframes individually; they can composite them offscreen and only display the complete frame.

The sync timeout field can be a number or <infinity>. Infinity can be represented by 0x7fffffff.

The termination condition given in the change_sync_timeout_and_termination field specifies how much longer, after the normal interframe delay has elapsed, the frame will endure. It can take the following values:

deterministic
The frame endures no longer than the normal interframe delay. Even though this is the default, a streaming encoder talking to a real-time decoder might write a FRAM with a termination condition of "deterministic" to force the display to be updated while the encoder decides its next move.

decoder-discretion
The decoder can lengthen the duration of the frame, but by no more than the timeout. A streaming decoder could take the opportunity to wait for its input buffer to fill to a comfortable level.

user-discretion
After the interframe delay has expired, the decoder should wait for permission from the user (e.g., via a keypress) before proceeding, but must wait no longer than the timeout. If the decoder cannot interact with the user, this condition degenerates into "decoder-discretion".

external-signal
After the interframe delay has expired, the decoder should wait for the arrival of a signal whose number matches a `sync_id`, but must wait no longer than the timeout.

The `sync_id` list can be omitted if the termination condition is not "external-signal".

When the `sync_id` list is changed, the number of `sync_id` entries is determined by the remaining length of the chunk data, divided by four. This number can be zero, which either inactivates the existing `sync_id` list for one frame or deletes it.

The initial values of the FRAM parameters are:

```
Framing mode              = 1
Subframe name             = <empty string>
Interframe delay          = 0
Left subframe boundary    = 0
Right subframe boundary   = frame width
Top subframe boundary     = 0
Bottom subframe boundary  = frame height
termination               = deterministic
Sync timeout              = 0x7fffffff (infinite)
Sync id                   = <empty list>
```

The DEFI chunk can be used to specify the placement of each image within the layer. The DEFI chunk can be used to specify clipping boundaries for each image. The subframe boundaries are only used for clipping, not for placement. Even when the left and top frame boundaries are nonzero, the image locations are measured with respect to the $\{0,0\}$ position in the display area. If the layers are transparent or do not cover the entire area defined by the subframe clipping boundaries, they are composited against the background defined by the BACK chunk, or against an application-defined background, if the BACK chunk is not present or is not recognized by the decoder. The background, as well as the images, is clipped to the subframe clipping boundaries. Any pixels outside the subframe clipping boundaries remain unchanged.

The frame_duration field gives the duration of display, which is the minimum time that must elapse from the beginning of displaying one frame until the beginning of displaying the next (or between images, when framing_mode=1). It is measured in "ticks" using the tick length determined from ticks_per_second defined in the MHDR chunk.

A viewer does not actually have to follow the procedure of erasing the screen, redisplaying the background, and recompositing the images against it, but what is displayed when the frame is complete must be the same as if it had. It is sufficient to redraw the parts of the display that change from one frame to the next.

The sync_id list provides a point at which the processor must wait for all pending processes to reach the synchronization point having the same sync_id before resuming, perhaps because of a need to synchronize a sound datastream (not defined in this specification) with the display, to synchronize stereo images, and the like. When the period defined by the sum of the frame_duration and the sync_timeout fields elapses, processing can resume even though the processor has not received an indication that other processes have reached the synchronization point.

Note that the synchronization point does not occur immediately, but at the end of the subframe that follows the FRAM chunk. If it is necessary to establish a synchronization point immediately, this can be done by using two consecutive FRAM chunks, the first setting a temporary frame_duration=0, sync_timeout, and sync_id, and the second establishing the synchronization point:

```
FRAM 2 0 1 1 0 1 0000 sync_timeout sync_id
FRAM 0 name
```

The identifier sync_id=0 is reserved to represent synchronization with a user input from a keyboard or pointing device. The sync_id values 1–255 are reserved to represent the corresponding ASCII letter, received from the keyboard (or a simulated keyboard), and values 256–1023 are reserved for future definition by this specification. If multiple channels (not defined in this specification) are not present, viewers can ignore other values appearing in the sync_id list.

3.3.3 TERM Termination action

The TERM chunk suggests how the end of the MNG datastream should be handled, when a MEND chunk is found. It contains either a single byte or ten bytes:

```
Termination
  action:      1 byte (unsigned integer)
                 0: Show the last frame indefinitely.
                 1: Cease displaying anything.
                 2: Show the first frame after the TERM
                    chunk.
                 3: Repeat the animation starting
                    immediately after the TERM chunk.

Action after
  iterations: 1 byte
```

```
                    0: Show the last frame indefinitely after
                       iteration_max iterations have been
                       done.
                    1: Cease displaying anything.
                    2: Show the first frame after the TERM
                       chunk.

                   This and the remaining fields must be
                   omitted if termination_action<=2, and
                   must be present otherwise.

Delay:             4 bytes (unsigned integer).  Delay, in
                   ticks, before repeating the animation.

Iteration max:     4 bytes (unsigned integer).  Maximum
                   number of times to repeat the animation.
```

The TERM chunk, if present, must appear either immediately after the MHDR chunk or immediately prior to a SEEK chunk. The TERM chunk is not considered to be a part of any segment for the purpose of determining the copy-safe status of any chunk. Only one TERM chunk is permitted in a MNG datastream.

Simple viewers and single-frame viewers can ignore the TERM chunk. It has been made critical only so MNG editors will not inadvertently relocate it.

3.4 SAVE and SEEK chunks

Simple decoders that only read MNG datastreams sequentially can safely ignore the SAVE and SEEK chunks.

3.5 Ancillary MNG chunks

This section describes ancillary MNG chunks. MNG-compliant decoders are not required to recognize and process them.

3.5.1 pHYs Physical pixel size

The MNG pHYs chunk is identical in syntax to the PNG pHYs chunk. When the framing mode is 2 or 4 in the FRAM chunk, it applies to complete MNG layers and not to the individual images within them. When the framing mode is 1 or 3, the MNG pHYs chunk provides a default value if the subsequent PNG datastream does not contain one.

The MNG top-level pHYs chunk can be nullified by a subsequent empty pHYs chunk appearing in the MNG top level.

3.5.2 PNG ancillary chunks

The namespace for MNG chunk names is separate from that of PNG. Only those PNG chunks named in this paragraph are also defined at the MNG top level. They have exactly the same syntax and semantics as when they appear in a PNG datastream:

- -[*iTXt*]
- tEXt
- tIME Same format as in PNG. Can appear at most once in the prologue segment (before the first SEEK chunk), and at most once per segment (between two consecutive SEEK chunks). In the prologue it indicates the last time any part of the MNG was modified. In a regular segment (between SEEK chunks), it indicates the last time that segment was modified.
- zTXt

A MNG editor that writes PNG datastreams should not include the top-level [*iTXt,*] tEXt, tIME, and zTXt chunks in the generated PNG datastreams.

4 The JPEG Network Graphics (JNG) Format

JNG (JPEG Network Graphics) is the lossy sub-format for MNG objects.

Note: This specification depends on the PNG Portable Network Graphics specification [PNG]. The PNG specification is available at the PNG home page,

```
http://www.cdrom.com/pub/png/
```

A JNG datastream consists of a header chunk (JHDR), JDAT chunks that contain a complete JPEG datastream, and optionally, IDAT chunks that contain a PNG-encoded grayscale image that is to be used as an alpha mask. Such a mask must have the same dimensions as the image itself. The JDAT and IDAT chunks can be interleaved. Some of the PNG ancillary chunks are also recognized in JNG datastreams.

While JNG is primarily intended for use as a sub-format within MNG, a single-image JNG datastream can be written in a standalone file. If so, the first eight bytes of a JNG datastream are

```
139 80 78 74 13 10 26 10
```

(decimal) which is similar to the PNG signature with "\213 J N G" instead of "\211 P N G" in bytes 0–3.

JNG is pronounced "Jing."

4.1 JNG critical chunks

This section specifies the critical chunks that are defined in the JNG format.

4.1.1 JHDR JNG header

The format of the JHDR chunk introduces a JNG datastream. It contains:

```
Width:      4 bytes (unsigned integer, range 0..65535).
Height:     4 bytes (unsigned integer, range 0..65535).
Color type: 1 byte
              8: Gray (Y).
             10: Color (YCbCr).
             12: Gray-alpha (Y-alpha).
             14: Color-alpha (YCbCr-alpha).

JDAT sample
  depth:    1 byte
              8: 8-bit samples and quantization tables.
             12: 12-bit samples and quantization
                 tables.
             20: 8-bit image followed by a 12-bit
                 image.

JDAT compression
  method:   1 byte
              8: ISO-10918-1 Huffman-coded baseline JPEG.

JDAT interlace
  method:   1 byte.
              0: Sequential JPEG, single scan.
              8: Progressive JPEG.

IDAT sample
  depth:    1 byte.
              0, 1, 2, 4, 8, or 16.

IDAT compression
  method:   1 byte.
              0: Zlib DEFLATE.

IDAT filter
  method:   1 byte.
              0: Adaptive (see PNG spec).

IDAT interlace
  method:   1 byte.
              0: Not interlaced.
```

The width, height, JDAT_sample_depth, JDAT_compression_method, and JDAT_interlace_method fields are redundant because equivalent information is also embedded in the JDAT datastream. They appear in the JHDR chunk for convenience. Their values must be identical to their equivalents embedded in the JDAT

chunk. We use four bytes in the width and height fields for similarity to MNG and PNG, and to leave room for future expansion, even though two bytes would have been sufficient.

When the `color_type` is 8 or 10 (no alpha channel), the last four bytes, which describe the IDAT data, must be set to zero. The `IDAT_sample_depth` must be nonzero when the alpha channel is present.

4.1.2 JDAT JNG image data

A JNG datastream must contain one or more JDAT chunks, whose data, when concatenated, forms a complete JNG JPEG datastream. JNG decoders are required to read all baseline JNG JPEG and eight-bit progressive JNG JPEG datastreams. Twelve-bit capability is not required.

JDAT chunks are like PNG IDAT chunks in that there may be multiple JDAT chunks, the data from which are concatenated to form a single datastream that can be sent to the decompressor. No chunks are permitted among the sequence of JDAT chunks, except for interleaved IDAT chunks. The ordering requirements of other ancillary chunks are the same with respect to JDAT as they are in PNG with respect to the IDAT chunk.

A JNG JPEG is a baseline, extended-sequential, or progressive JPEG as defined by JPEG Part 1 [ISO/IEC-10918-1]. JNG uses only JFIF-compatible [JFIF] component interpretations, and imposes a few additional restrictions that reflect limitations of many existing JPEG implementations. In particular, only Huffman entropy coding is permitted.

Actually, a JNG may contain two separate JNG JPEG datastreams (one eight-bit and one twelve-bit), each contained in a series of JDAT chunks, and separated by a JSEP chunk (see the JSEP chunk specification below, Paragraph 5.1.4). Decoders that are unable to handle twelve-bit datastreams are allowed to display the eight-bit datastream instead, if one is present.

The core of the JNG JPEG definition is baseline JNG JPEG, which is JPEG Part 1's definition of baseline JPEG further restricted by JFIF restrictions and JNG-specific restrictions. JPEG, which is also defined in JPEG Part 1 and has JNG-specific restrictions.

- Baseline JNG JPEG restrictions

  A baseline JPEG according to JPEG Part 1 is DCT-based (lossy) sequential JPEG, using 8-bit sample precision and Huffman entropy coding, with the following further restrictions:

  - Quantization table precision must be 8 bits for baseline JPEG.
  - Huffman code tables can have table numbers 0 and 1 only.

  The SOF marker type for baseline JPEG is SOF0.

  JDAT datastreams must always follow "interchange JPEG" rules: all necessary quantization and Huffman tables must be included in the datastream; no tables can be omitted.

- JFIF-compatible restrictions

  The image data is always stored left-to-right, top-to-bottom.

The encoded data shall have one of the two color space interpretations allowed by the JFIF specification:

- Grayscale: a single component representing luminance, ranging from 0 for black to 255 for white (or 0 to 4095 when dealing with twelve-bit data). This component shall have JPEG component identifier 1.
- YCbCr: three components representing luminance, chroma blue, and chroma red, in that order. The components shall be assigned JPEG component identifiers 1, 2, 3 respectively. YCbCr is defined as a linear transformation from RGB color space:

  ```
  Y = Luma_red*R + Luma_green*G + Luma_blue*B
  Cb = (B - Y) / (2 - 2*Luma_blue) + Half_scale
  Cr = (R - Y) / (2 - 2*Luma_red)  + Half_scale
  ```

  By convention, the luminance coefficients are always those defined by CCIR Recommendation 601-1:

  ```
  Luma_red   = 0.299
  Luma_green = 0.587
  Luma_blue  = 0.114
  ```

  The constant Half_scale is 128 when dealing with eight-bit data, 2048 for twelve-bit data. With these equations, Y, Cb, and Cr all have the same range as R, G, and B: 0 to 255 for eight-bit data, 0 to 4095 for twelve-bit data.

  The JFIF convention for YCbCr differs from typical digital television practice in that no headroom/footroom is reserved: the coefficient values range over the full available 8 or 12 bits.

  Intercomponent sample alignment shall be such that the first (upper leftmost) samples of each component share a common upper left corner position. This again differs from common digital TV practice, in which the first samples share a common center position. The JFIF convention is simpler to visualize: subsampled chroma samples always cover an integral number of luminance sample positions, whereas with co-centered alignment, chroma samples only partially overlap some luminance samples.

- Additional JNG restrictions

  JNG imposes three additional restrictions not found in the text of either JPEG Part 1 or the JFIF specification:

  - The sampling factors for YCbCr images must be one of these sets:
    * 1h1v,1h1v,1h1v (also called 4:4:4 or 1x1 sampling)
    * 2h1v,1h1v,1h1v (also called 4:2:2 or 2x1 sampling)
    * 2h2v,1h1v,1h1v (also called 4:2:0 or 2x2 sampling)
    * 1h2v,1h1v,1h1v (also called 1x2 sampling)

    In other words, the chroma components may be downsampled 2:1 or 1:2 horizontally or vertically relative to luminance, or they may be left full size. These four sampling ratios are the only ones

supported by a wide spectrum of implementations (1x2 is relatively uncommon, and is usually the result of a lossless rotation of a 2x1 sampling).

For grayscale images, the sampling factors are irrelevant according to a strict reading of JPEG Part 1. Hence decoder authors should accept any sampling factors for grayscale. However, we recommend that encoders always emit sampling factors 1h1v for grayscale, since some decoders have been observed to malfunction when presented with other sampling factors.

– There must be only one scan in an image: that is, YCbCr images must be fully interleaved. There is little advantage to be gained by encoding a baseline image in multiple scans, and many baseline decoders do not support multiple scans at all.

– The DNL (Define Number of Lines) marker is prohibited. The image height must always be specified accurately in the SOFn marker and in the JHDR chunk.

- Recommended progressive JPEG subset

  For JNG progressive JPEG datastreams, the JPEG process is progressive Huffman coding (SOF marker type SOF2) rather than baseline (SOF0). All JNG-compliant decoders must support full progression, including both spectral-selection and successive-approximation modes, with any sequence of scan progression parameters allowed by the JPEG Part 1 standard.

  Otherwise, all the restrictions listed above apply, except these:

  – Multiple-scan support is obviously required for progressive JPEG.

  – Huffman table numbers up to 3 (the full JPEG limit) may be used, since the baseline two-table limit is unlikely to be needed by any decoder that can handle progressive JPEG.

  We require full progression support since relatively little code savings can be achieved by subsetting the JPEG progression features. In particular, successive approximation offers significant gains in the visual quality of early scans. Omitting successive-approximation support from a decoder does not save nearly enough code to justify restricting JNG progressive encoders to spectral selection only.

  No particular progressive scan sequence is specified or recommended by this specification. Not enough experience has been gained with progressive JPEG to warrant making such a recommendation. To allow for future experimentation with scan sequences, decoders are expected to handle any JPEG-legal sequence. Again, the code savings that might be had by making restrictive assumptions are too small to justify a limitation.

  When the JSEP chunk is present, both images must be progressive if one of them is progressive.

- Recommended 12-bit JPEG subset

  JNG JPEGs may optionally use 12-bit sample precision as defined in JPEG Part 1.

  For a sequential image, the SOF marker type must be SOF1 (extended sequential) not SOF0, and the baseline restriction of two Huffman tables is removed. Also, the encoder may use either 8-bit or 16-bit quantization tables. All other JNG baseline restrictions still apply. It is recommended that JNG encoders not use extended-sequential mode except to encode 12-bit data.

For a progressive image, the only difference between 8-bit and 12-bit modes is that the sample precision is 12 bits and the encoder may use either 8-bit or 16-bit quantization tables. All other JNG restrictions still apply.

4.1.3 IDAT JNG alpha data

This chunk is exactly like the IDAT chunk in a PNG grayscale image, except that it is interpreted as an alpha mask to be applied to the image data from the JDAT chunks. The alpha channel, if present, can have sample depths 1, 2, 4, 8, or 16. The IDAT chunks can be interleaved with the JDAT chunks (see Recommendations for Encoders: JNG interleaving below). No other chunk type can appear among the sequence of IDAT and JDAT chunks. No other chunk type can appear between the sequences of IDAT and JDAT chunks when they are not interleaved. The samples in the IDAT must be presented in noninterlaced order, left to right, top to bottom. As in PNG, zero means fully transparent and $2^{\mathrm{IDAT_sample_depth}} - 1$ means fully opaque.

The IDAT chunks must precede the JSEP chunk, if the JSEP chunk is present. Minimal viewers that ignore the twelve-bit JDAT chunks must read the IDAT chunks and apply the alpha samples to the eight-bit image that is contained in the JDAT chunks that precede the JSEP chunk. Viewers that skip the eight-bit JDAT chunks must read the IDAT chunks that precede the JSEP chunk and apply the alpha samples to the twelve-bit image that is contained in the JDAT chunks that follow the JSEP chunk.

The JNG IEND chunk is identical to its counterpart in PNG. Its data length is zero, and it serves to mark the end of the JNG datastream.

4.1.4 JSEP 8-bit/12-bit image separator

The JSEP chunk is empty.

A JSEP chunk must appear between the JDAT chunks of an eight-bit datastream and those of a twelve-bit datastream, when `JDAT_sample_depth=20` in the JHDR chunk. The eight-bit datastream must appear first. Both images must have the same width, height, color type, compression method, and interlace type. Viewers can choose to display one or the other image, but not both.

4.2 JNG ancillary chunks

Some PNG ancillary chunks can also appear in JNG datastreams, and are used for the same purposes as described in the PNG specification:

If the bKGD chunk is present, it must be written as if it were written for a PNG datastream with sample_depth=8. It has one 2-byte entry for grayscale JNGs and three 2-byte entries for color JNGs. The first (most significant) byte of each entry must be 0.

The following chunks have exactly the same meaning and have the same format as given in the PNG specification: cHRM, gAMA, iCCP, sRGB, pHYs, oFFs, [*iTXt,*] tEXt, tIME, and zTXt.

The PNG PLTE, hIST, pCAL, sBIT, and tRNS chunks are not defined in JNG.

When cHRM, gAMA, iCCP, or sRGB are present, they provide information about the color space of the decoded JDAT image, and they have no effect on the decoded alpha samples from the IDAT chunks. Any viewer that processes the gAMA chunk must also recognize and process the sRGB chunk. It can treat it as if it were a gAMA chunk containing the value .45455 and it can ignore its "intent" field.

The chunk copying and ordering rules for JNG are the same as those in PNG, except for the fact that JDAT and IDAT chunks can be interleaved.

5 Chunk Copying Rules

The chunk copying rules for MNG are the same as those in PNG, except that a MNG editor is not permitted to move unknown chunks across any of the following chunks:

- SAVE
- SEEK
- IHDR
- JHDR
- IEND
- BASI

The copy-safe status of an unknown chunk is determined from the chunk name, just as in PNG. If bit 5 of the first byte of the name is 0 (Normally corresponding to an uppercase ASCII letter), the unknown chunk is critical and cannot be processed or copied. If it is 1 (usually corresponding to a lowercase ASCII letter), the unknown chunk is ancillary and its copy-safe status is determined by bit 5 of the fourth byte of the name, 0 meaning copy-unsafe and 1 meaning copy-safe.

If an editor makes changes to the MNG datastream that render unknown chunks unsafe-to-copy, this does not affect the copy-safe status of any chunks beyond the next SEEK chunk or prior to the previous one. However, if it makes such changes prior the SAVE chunk, this affects the copy-safe status of all top-level unknown chunks in the entire MNG datastream.

Changes to the MHDR chunk do not affect the copy-safe status of any other chunk.

The SAVE, SEEK, and TERM chunks are not considered to be a part of any segment. Changes to the data in the SAVE or SEEK chunks do not affect the copy-safe status of any other chunks. Adding or removing a SEEK chunk affects the copy-safe status of unknown chunks in the newly-merged or newly-separated segments. Adding, removing, or changing the TERM chunk has no effect on the copy-safe status of any chunk.

As in PNG, unsafe-to-copy ancillary chunks in the top-level MNG datastream can have ordering rules only with respect to critical chunks. Safe-to-copy ancillary chunks in the top-level MNG datastream can have ordering rules only with respect to the SAVE, SEEK, IHDR-IEND, JHDR-IEND, and BASI-IEND sequences,

or with respect to any other critical "header-end" sequence that might be defined in the future that could contain IDAT or similar chunks.

The copying rules for unknown chunks inside IHDR-IEND, BASI-IEND, and JHDR-IEND sequences are governed by the PNG and JNG specifications, and any changes inside such sequences have no effect on the copy-safe status of any top-level MNG chunks.

6 Minimum Requirements for MNG-Compliant Viewers

This section specifies the minimum level of support that is expected of MNG-compliant decoders, and provides recomendations for viewers that will support slightly more than the minimum requirements. All critical chunks must be recognized, but some of them can be ignored after they have been read and recognized. Ancillary chunks can be ignored, and do not even have to be recognized.

Anything less than this level of support would require subsetting. Applications that provide less than minimal support should check the MHDR "simplicity profile" for the presence of features that they are unable to support. We are allowing minimal decoders to skip twelve-bit JNGs because those are likely to be rarely encountered and used only for special purposes.

6.1 Required PNG chunk support

IHDR, PLTE, IDAT, IEND
: All PNG critical chunks must be fully supported. All values of `color_type`, `bit_depth`, `compression_method`, `filter_method` and `interlace_method` must be supported (interlacing, as in PNG, need not necessarily be displayed on-the-fly; the image can be displayed after it is fully decoded). The alpha-channel must be supported, at least to the degree that fully opaque pixels are opaque and fully transparent ones are transparent. It is recommended that alpha be fully supported.

tRNS
: The PNG tRNS chunk, while it is an ancillary chunk, must be supported in MNG-compliant viewers, at least to the degree that fully opaque pixels are opaque and fully transparent ones are transparent. It is recommended that alpha data from the tRNS chunk be fully supported in the same manner as alpha data from an RGBA image or a JNG with an alpha channel contained in IDAT chunks.

Other PNG ancillary chunks
: Ancillary chunks other than PNG tRNS can be ignored, and do not even have to be recognized.

Color management
: It is highly recommended that decoders support at least the gAMA chunk to allow platform-independent color rendering.

6.2 Required JNG chunk support

JHDR, JDAT, IDAT, JSEP, IEND
All JNG critical chunks must be fully supported. All values of `color_type`, `bit_depth`, `compression_method`, `filter_method` and `interlace_method` must be supported (interlacing, as in PNG, need not necessarily be displayed on-the-fly; the image can be displayed after it is fully decoded). The alpha-channel must be supported, at least to the degree that fully opaque pixels are opaque and fully transparent ones are transparent. It is recommended that alpha be fully supported. Bit 3 of the simplicity profile can be used to promise that none of these chunks are present.

JNG ancillary chunks
All JNG ancillary chunks can be ignored, and do not even have to be recognized.

JDAT sample depth
Only `sample_depth=8` must be supported. The JSEP chunk must be recognized and must be used by minimal decoders to select the eight-bit version of the image, when both eight-bit and twelve-bit versions are present, as indicated by `JDAT_sample_depth=20` in the JHDR chunk. When `JDAT_sample_depth=12`, minimal decoders are not obligated to display anything, but should display an empty transparent rectangle of the width and height specified in the JHDR chunk. This can be done by processing the JNG as though a viewable transparent `BASI` object had appeared:

```
BASI width height 1 4 0 0 0 0 00 00 00 00 1
IEND
```

6.3 Required MNG chunk support

MHDR
The `ticks_per_second` must be supported by animation viewers. The simplicity profile, frame count, layer count, and nominal play time can be ignored. Decoders that provide less than minimal support can use the simplicity profile to identify datastreams that they are incapable of processing.

MEND
The MEND chunk must be recognized but does not require any processing other than completing the last frame.

LOOP, ENDL
The LOOP chunk and its ENDL chunk can be ignored (bit 1 of the simplicity profile that this is true for all loops).

DEFI, BASI, BACK
Must be fully supported.

FRAM
The `framing_mode` and clipping parameters must be supported. The `interframe_delay`

must be supported except by single-frame viewers. The `sync_id` and `sync_timeout` data can be ignored.

SAVE, SEEK, TERM
Can be ignored.

7 Recommendations for Encoders

The following recommendations do not form a part of the specification.

7.1 Use a common color space

It is a good idea to use a single color space for all of the layers in an animation, where speed and fluidity are more important than exact color rendition. This is best accomplished by defining a single color space at the top level of MNG, using gAMA and cHRM chunks, and either an sRGB or iCCP chunk, and removing any color space chunks from the individual images after converting them to the common color space.

When the encoder converts all images to a single color space before putting them in the MNG datastream, this will allow decoders to improve the speed and consistency of the display.

For single-frame MNG datastreams, however, where decoding speed is less important and exact color rendition might be more important, it is best to leave the images in their original color space, as recommended in the PNG specification, to avoid any loss of data due to conversion, and to retain the individual color space chunks if the images have different color spaces.

7.2 Use the right framing mode

Always use framing mode 1 or 2 when all of the images are opaque. This avoids unnecessary screen clearing, which can cause flickering.

7.3 Including optional index in SAVE chunk

Authors of MNG files that are intended for transmission over a network should consider whether it is more economical for the client to rebuild the index from scratch than it is to transmit it. Web pages that are likely to be downloaded over slow lines, and whose clients are unlikely to use the index anyway, generally should have empty SAVE chunks. No information is lost by deleting the index, because the MNG datastream contains all of the information needed to build the index. If an application does build an index, and the file is going to be kept as a local file, the application should replace the empty SAVE chunk with one containing the index. See above (Paragraph 4.4.1).

7.4 Interleaving JDAT and IDAT chunks

When a JNG datastream contains an alpha channel, and the file is intended for transmission over a network, it is useful to interleave the IDAT and JDAT chunks. In the case of sequential JPEG, the interleaving should be arranged so that the alpha data arrives more or less in sync with the color data for the scanlines. In the case of progressive JPEG, the alpha data should be interleaved with the first JPEG pass, so that *all* of the alpha data has arrived before beginning the second JPEG pass.

8 Recommendations for Decoders

8.1 Note on compositing

The PNG specification gives a good explanation of how to composite a partially transparent image over an opaque image, but things get more complicated when both images are partially transparent.

Pixels in PNG and JNG images are represented using gamma-encoded RGB (or gray) samples along with a linear alpha value. Alpha processing can only be performed on linear samples. This chapter assumes that R, G, B, and A values have all been converted to real numbers in the range [0..1], and that any gamma encoding has been undone.

For a top pixel {Rt,Gt,Bt,At} and a bottom pixel {Rb,Gb,Bb,Ab}, the composite pixel {Rc,Gc,Bc,Ac} is given by:

```
Ac = 1 - (1 - At)(1 - Ab)
if (Ac != 0) then
  s = At / Ac
  t = (1 - At) Ab / Ac
else
  s = 0.0
  t = 1.0
endif
Rc = s Rt + t Rb
Gc = s Gt + t Gb
Bc = s Bt + t Bb
```

When the bottom pixel is fully opaque (Ab = 1.0), the function reduces to:

```
Ac = 1
Rc = At Rt + (1 - At) Rb
Gc = At Gt + (1 - At) Gb
Bc = At Bt + (1 - At) Bb
```

When the bottom pixel is not fully opaque, the function is much simpler if premultiplied alpha is used. A pixel that uses non-premultiplied alpha can be converted to premultiplied alpha by multiplying R, G, and B by A.

For a premultiplied top pixel {Rt,Gt,Bt,At} and a premultiplied bottom pixel {Rb,Gb,Bb,Ab}, the premultiplied composite pixel {Rc,Gc,Bc,Ac} is given by:

```
Ac = 1 - (1 - At)(1 - Ab)
Rc = Rt + (1 - At) Rb
Gc = Gt + (1 - At) Gb
Bc = Bt + (1 - At) Bb
```

As mentioned in the PNG specification, the equations become much simpler when no pixel has an alpha value other than 0.0 or 1.0, and the RGB samples need not be linear in that case.

8.2 Decoder handling of fatal errors

When a fatal error is encountered, such as a bad CRC or an unknown critical MNG chunk, minimal viewers that do not implement the SAVE/SEEK mechanism should simply abandon the MNG datastream. More capable MNG viewers should attempt to recover gracefully by abandoning processing of the segment and searching for a SEEK chunk. If such errors occur before the SAVE chunk is reached, the viewer should abandon the MNG datastream.

When an error occurs within a image datastream, such as an unknown critical PNG chunk or a missing parent object where one was required, only that image should be abandoned and the associated object should be discarded. If a bad CRC is found, indicating a corrupted datastream, the entire segment should be abandoned, as above.

MNG editors, on the other hand, should be more strict and reject any datastream with errors unless the user intervenes.

8.3 Decoder handling of interlaced images

Decoders are required to be able to interpret datastreams that contain interlaced PNG images, but are only required to display the completed frames; they are not required to display the images as they evolve. Viewers that are decoding datastreams coming in over a slow communication link might want to do that, but MNG authors should not assume that the frames will be displayed in other than their final form.

8.4 Decoder handling of palettes

When a PLTE chunk is received, it only affects the display of the PNG datastream that includes it, and any subsequent Delta-PNG datastreams that depend on it.

If a frame contains two or more images, the PLTE chunk in one image does not affect the display of the other.

A composite frame consisting only of indexed-color images should not be assumed to contain 256 or fewer colors, since the individual palettes do not necessarily contain the same set of colors.

8.5 Behavior of single-frame viewers

Viewers that can only display a single frame must display the first frame that they encounter.

8.6 Clipping

MNG-Lite provides three types of clipping, in addition to any clipping that might be required due to the physical limitations of the display device.

Frame width and frame height

The `frame_width` and `frame_height` are defined in the MHDR chunk and cannot be changed.

Decoders can use these parameters to establish the size of a window in which to display the MNG frames. When the `frame_width` or `frame_height` exceeds the physical dimensions of the display hardware, the contents of the area outside those dimensions is undefined. If a viewer chooses, it can create "scroll bars" or the like, to enable persons to pan and scroll to the offscreen portion of the frame. If this is done, then the viewer is responsible for maintaining and updating the offscreen portion of the frame.

In the case of a MNG datastream that consists of a PNG or JNG datastream, with the PNG or JNG signature, the `frame_width` and `frame_height` are defined by the `width` and `height` fields of the IHDR (or JHDR) chunk.

Subframe clipping boundaries

The subframe clipping boundaries are optionally defined in the FRAM chunk, and cannot be changed within a subframe. When the framing mode is 3 or 4, viewers must, prior to displaying each subframe, clear the area within the subframe clipping boundaries to the background color, and display the background image if one has been defined. Viewers must not change any pixels outside the subframe boundaries; encoders must be able to rely on the fact that the part of the display that is outside the subframe clipping boundaries (but inside the area defined by `frame_width` and `frame_height`) will remain on the display from frame to frame without being explicitly redisplayed. See Example 8, which displays a large background image once, and then, in each frame, only redisplays the portion of the background surrounding the moving sprite.

Image clipping boundaries

The image clipping boundaries are defined in the DEFI chunk. They are associated with individual objects, not with the subframes, and they can be changed within a subframe. They are useful for exposing only a portion of an image in a frame.

9 Recommendations for Editors

9.1 Editing datastreams with optional index

Editors must recreate or delete the optional SAVE chunk index whenever they make any change that affects the offsets of chunks following the portion of the datastream that is changed. If the changes do not involve the addition, deletion, or relocation of segments, frames, and images, then it is sufficient to zero out the offsets.

The SAVE chunk is not considered to be in any MNG segment, so changing it has no effect on the copy-safe status of unknown chunks in any other part of the MNG datastream.

When the SAVE chunk is expanded to include an index, all chunks that follow will have their offsets changed by an amount equal to the change in the length of the data segment of the SAVE chunk, so the offset table will have to be adjusted accordingly. If a SAVE chunk is already present with zero offsets, the correct offsets can be written without adjustment.

10 Miscellaneous Topics

10.1 File name extension

On systems where file names customarily include an extension signifying file type, the extension `.mng` is recommended for MNG files. Lowercase `.mng` is preferred if file names are case-sensitive. The extension `.jng` is recommended for JNG files.

10.2 Internet media type

When and if the MNG format becomes finalized, the MNG authors intend to register `video/mng` as the Internet Media Type for MNG [RFC-2045], [RFC-2048].

At the date of this document, the media type registration process had not been started. It is recommended that implementations also recognize the interim media type `video/x-mng`.

Although we define a standalone JNG format, we recommend that such files be used only temporarily while compiling or disassembling MNG datastreams. We do not intend to register an Internet Media Type for JNG files.

10.3 Uniform Resource Identifier (URI)

Segments, subframes, and objects are externally accessible via named SEEK, and FRAM chunk names. They can be referred to by URI, as in

```
SRC=file.mng#segment_name
SRC=file.mng#subframe_name
SRC=file.mng?segment_name#segment_name
```

When the URI specializer ("#" or "?") is "#", and the fragment identifier (the string following the specializer) is the name of a segment, i.e., a named SEEK chunk, the viewer should display the animation from the beginning of the named segment up to the next segment. When it refers to a subframe or an image, i.e., a named FRAM chunk, it should display the single frame (as it exists when the next FRAM chunk is encountered) or image that is identified by the fragment identifier. If the SAVE chunk is present and contains the optional index, this can help the client find the needed segment quickly.

When the URI specializer is "?" (server side query), the "query component" is the string following the "?" specializer and up to but not including the "#" if the "#" specializer is also present. The server should find the segment that is named in the query component or the segment that contains the subframe or image named in the query component, and it should return a datastream consisting of:

- all chunks prior to the SAVE chunk,
- an empty SAVE chunk,
- the SEEK chunk for the segment being returned,
- all chunks in the segment, and
- a MEND chunk.

If no SAVE chunk is present, the server must simply return the entire MNG datastream. Servers that are unwilling to parse the MNG datastream and are unconcerned about bandwidth can return the entire MNG datastream even when the SAVE chunk is present. Authors should defend against this behavior by including both a query and a fragment in the URI even when a segment is being requested.

The client can process this as a complete MNG datastream, either displaying the entire segment, if no fragment identifier is present, or extracting the segment, subframe or image that is named in a fragment identifier and displaying it, if a fragment identifier is present (a fragment identifier must be present if a frame or image is being requested).

A part of the MNG datastream can also be requested by timecode, as in

```
SRC=file.mng#clock(10s-20s)
SRC=file.mng#clock(0:00-0:15)
SRC=file.mng?clock(0:00-0:15#clock=0:00-0:15)
```

or by frame number, as in

```
SRC=file.mng#frame(10)
SRC=file.mng#frames(30-60)
SRC=file.mng?frames(30-60#frames=30-60)
```

The timecode must consist of starting and ending clock values, as defined in the W3C SMIL recommendation, separated by a hyphen (ASCII code 45).

When the URI specializer is "#", the viewer should play that part of the animation beginning and ending at the requested times, measuring from zero time at the beginning of the MNG datastream, or beginning and ending with the specified frame numbers.

When the URI specializer is "?", the server can send the entire MNG datastream, or, preferably, it should construct a complete MNG file containing:

- the chunks prior to the SAVE chunk,
- the SAVE chunk itself with an optional index, and
- the one or more consecutive set of segments, with their SEEK chunks, that contain the animation beginning and ending at the requested times, or frame numbers, at the proper framing rate.

If the server does not send the entire MNG datastream, and the first segment after the SAVE chunk is not sent, the optional index must be written even if it does not exist in the source file. The index must contain at least one "type 0" entry that gives the nominal start time and frame number for the first segment that is sent after the SAVE chunk. The offset field can be set to zero and the segment name can be omitted.

The query component should always be repeated as a fragment identifier, so clients can find the requested item in case the server sends more than what was requested.

MNG datastreams should not contain segment, subframe, or image names that begin with the case-insensitive strings "clock(", "frame(", or "frames(", which are reserved for use in URI queries and fragments.

See [RFC-2396] and the W3C SMIL recommendation at http://www.w3.org/TR/.

11 References

[ISO/IEC-10918-1]

International Organization for Standardization and International Electrotechnical Commission, "Digital Compression and Coding of Continuous-tone Still Images, Part 1: Requirements and guidelines" ISO/IEC IS 10918-1, ITU-T T.81.
See also Pennebaker, William B., and Joan L. Mitchell, "JPEG : Still Image Data Compression Standard" Van Nostrand Reinhold, ISBN:0442012721, September 1992

[JFIF]

C-Cube Microsystems, "JPEG File Interchange Format, Version 1.02", September 1992.

[PNG]

Boutell, T., et. al., "PNG (Portable Network Graphics Format) Version 1.0", RFC 2083, ftp://ftp.isi.edu/in-notes/rfc2083.txt also available at ftp://swrinde.nde.swri.edu/pub/png/documents/. This specification has also been published as a W3C

Recommendation, which is available at
http://www.w3.org/TR/REC-png.html.

See also the PNG-1.1 draft:
Randers-Pehrson, G., et. al., "PNG (Portable Network Graphics Format) Version 1.1", which is available at
ftp://swrinde.nde.swri.edu/pub/png/documents/.

[PNG-EXT]
Extensions to the PNG 1.1 Specification,
ftp://swrinde.nde.swri.edu/pub/png/documents/pngext-*.

[RFC-2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119/BCP 14, Harvard University, March 1997.

[RFC-2045]
Freed, N., and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, Innosoft, First Virtual, November 1996.
ftp://ftp.isi.edu/in-notes/rfc2045.txt

[RFC-2048]
Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, Innosoft, MCI, USC/Information Sciences Institute, November 1996.
ftp://ftp.isi.edu/in-notes/rfc2048.txt

[RFC-2396]
Berners-Lee, T., R. Fielding, U. C. Irvine, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, MIT/LCS, Xerox Corporation, University of Minnesota, August 1998.
ftp://ftp.isi.edu/in-notes/rfc2396.txt

12 Security Considerations

Security considerations are addressed in the basic PNG specification.

Some people may experience epileptic seizures when they are exposed to certain kinds of flashing lights or patterns that are common in everyday life. This can happen even if the person has never had any epileptic seizures. All graphics software and file formats that support animation and/or color cycling make it possible to encode effects that may induce an epileptic seizure in these individuals. It is the responsibility of authors and software publishers to issue appropriate warnings to the public in general and to animation creators in particular.

No known additional security concerns are raised by this format.

13 Appendix: Examples

We use the "#" character to denote commentary in these examples; such comments are not present in actual MNG datastreams.

13.1 Example 1: A single image

The simplest MNG datastream is a single-image PNG datastream. The simplest way to create a MNG from a PNG is:

```
copy file.png file.mng
```

The resulting MNG file looks like:

```
\211 P N G \r \n ^z \n  # PNG signature.
IHDR 720 468 8 0 0 0 0  # Width and Height, etc.
IDAT ...
IEND
```

13.2 Example 2: Very simple movie

This example demonstrates a very simple movie, such as might result from directly converting an animated GIF that contains a simple series of full-frame images:

```
\212 M N G \r \n ^z \n  # MNG signature.
MHDR 256 300   # Width and height.
     30        # 30 ticks per second.

TERM 3 0 120 10   # When done, repeat from TERM 10 times.

FRAM 1 0 2 0 0 0 30 # Set framing_mode=1 (because the
        #  images are opaque) and frame_duration to 1 sec.

DEFI 1 IHDR ...  IDAT ...  IEND # Eight PNG datastreams
DEFI 2 IHDR ...  IDAT ...  IEND # are read and stored as
DEFI 3 IHDR ...  IDAT ...  IEND # abstract images, and
DEFI 4 IHDR ...  IDAT ...  IEND # are displayed as they
DEFI 5 IHDR ...  IDAT ...  IEND # are read.
DEFI 6 IHDR ...  IDAT ...  IEND
DEFI 7 IHDR ...  IDAT ...  IEND
DEFI 8 IHDR ...  IDAT ...  IEND

MEND
```

13.3 Example 3: Simple slideshow

```
\212 M N G \r \n ^z \n  # MNG signature.
MHDR 720 468   # Width and height.
     1         # 1 tick per second.
FRAM 1 0 2 2 0  2 0 600 0 # Set frame_duration to 0,
  # sync_timeout to 600 sec, and sync_id list to {0}.
SAVE
SEEK "Briefing to the Workforce"
IHDR ...  IDAT ...  IEND  # DEFI 0, visible, abstract
SEEK "Outline"            # is implied.
IHDR ...  IDAT ...  IEND
SEEK "Our Vision"       IHDR ...  IDAT ...  IEND
SEEK "Our Mission"      IHDR ...  IDAT ...  IEND
SEEK "Downsizing Plans" IHDR ...  IDAT ...  IEND
MEND
```

13.4 Example 4: A more storage-efficient slideshow

14 Credits

Editor

- Glenn Randers-Pehrson randeg @ alumni.rpi.edu

Contributors

Contributors' names are presented in alphabetical order:

- Mark Adler, madler @ alumni.caltech.edu
- Thomas Boutell, boutell @ boutell.com
- John Bowler, jbowler @ acm.org
- Christian Brunschen, cb @ df.lth.se
- Glen Chapman, glenc @ clark.net
- Adam M. Costello, amc @ cs.berkeley.edu
- Lee Daniel Crocker, lee @ piclab.com
- Peter da Silva, peter @ starbase.neosoft.com
- Andreas Dilger, adilger @ enel.ucalgary.ca
- Oliver Fromme, oliver @ fromme.com
- Jean-loup Gailly, jloup @ gzip.org
- Chris Herborth, chrish @ qnx.com
- Alex Jakulin, Aleks.Jakulin @ snet.fri.uni-lj.si
- Gerard Juyn, gjuyn @ xs4all.nl
- Neal Kettler, neal @ westwood.com
- Tom Lane, tgl @ sss.pgh.pa.us
- Alexander Lehmann, alex @ hal.rhein-main.de
- Chris Lilley, chris @ w3.org
- Dave Martindale, davem @ cs.ubc.ca
- Carl Morris, msftrncs @ htcnet.com
- Owen Mortensen, ojm @ acm.org

- Josh M. Osborne, stripes @ va.pubnix.com
- Keith S. Pickens, ksp @ swri.edu
- Glenn Randers-Pehrson, randeg @ alumni.rpi.edu
- Nancy M. Randers-Pehrson, randeg @ alumni.rpi.edu
- Greg Roelofs, newt @ pobox.com
- Willem van Schaik, willem @ schaik.com
- Guy Schalnat, gschal @ infinet.com
- Paul Schmidt, pschmidt @ photodex.com
- Smarry Smarasderagd, smar @ reptiles.org
- Thomas R. Tanner, ttehtann @ argonet.co.uk
- Guido Vollbeding, guivol @ esc.de
- Tim Wegner, twegner @ phoenix.net
- Alaric B. Williams, png @ abwillms.demon.co.uk

Trademarks

- GIF is a service mark of CompuServe Incorporated.
- PostScript is a trademark of Adobe Systems.
- X Window System is a trademark of the Massachusetts Institute of Technology.

Document source

This document was built from the file `mng-master-19990215` on 19 February 1999.

Copyright Notice

Copyright © 1998, 1999 by: Glenn Randers-Pehrson

This specification is being provided by the copyright holder under the following license. By obtaining, using and/or copying this specification, you agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and distribute this specification for any purpose and without fee or royalty is hereby granted, provided that the full text of this **NOTICE** appears on *ALL* copies of the specification or portions thereof, including modifications, that you make.

THIS SPECIFICATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDER MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SPECIFICATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDER WILL BEAR NO LIABILITY FOR ANY USE OF THIS SPECIFICATION.

The name and trademarks of copyright holder may *NOT* be used in advertising or publicity pertaining to the specification without specific, written prior permission. Title to copyright in this specification and any associated documentation will at all times remain with copyright holder.

End of MNG-Lite Specification.