PNG Special−Purpose Public Chunks

Revision date: 8 November, 1995

The latest versions of this document and related information can always be found at the PNG FTP archive site, ftp.uu.net:/graphics/png/. The maintainers of the PNG specification can be contacted by e−mail at png−info@uunet.uu.net.

At present, this document is available on the World Wide Web from http://sunsite.unc.edu/boutell/pngextensions.html, but this location may not be as permanent as the ones above.

Introduction

This document describes public chunk types for use in PNG (Portable Network Graphics) images. The basic PNG specification is available from the PNG FTP archive mentioned above, or via WWW from http://sunsite.unc.edu/boutell/png.html.

Chunks described here are expected to be less widely supported than those defined in the basic specification. However, application authors are encouraged to use these chunk types whenever appropriate for their applications. Additional chunk types may be proposed for inclusion in this list by contacting the PNG specification maintainers at png−info@uunet.uu.net.

This document also describes data representations that do not occur in the core PNG format, but are used in one or more special−purpose chunks. New chunks should use these representations whenever applicable, in order to maximize portability and simplify decoders.

Data Representation

Floating−Point Values

The core of PNG does not use floating−point numbers anywhere; it uses integers or, where applicable, fixed−point fractional values. However, special−purpose chunks may need to represent values that do not fit comfortably in fixed−point notation. The textual floating−point notation defined here is recommended for use in all such cases. This representation is simple, has no a−priori limits on range or precision, and is portable across all machines.

A floating−point value in this notation is represented by an ASCII text string in a standardized decimal floating−point format. The string is variable−length and should be terminated by a null (zero) character unless it is the last item in its chunk. The string consists of an optional sign ('+' or '−'), an integer part, a fraction part beginning with a decimal point ('.'), and an exponent part beginning with an 'E' or 'e' and optional sign. The integer, fraction, and exponent parts each contain one or more digits (ASCII '0' to '9'). Either the integer part or the fraction part, but not both, may be omitted. A decimal point is allowed, but not required, if there is no fraction part. The exponent part may be omitted. No spaces or any other character besides those specified may appear.

Note in particular that C−language F and L suffixes are not allowed, the string "." is not allowed as a shorthand for 0 as in some other programming languages, and no commas or underscores are allowed. This format should be easily readable in all programming environments.

Summary of Special−Purpose Chunks

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

```
Name  Multiple  Ordering constraints
        OK?

oFFs    No      Before IDAT
sCAL    No      Before IDAT
gIFg    Yes     None
gIFt    Yes     None
gIFx    Yes     None
fRAc    Yes     None
```

Chunk Descriptions

`oFFs` Image Offset

The `oFFs` chunk gives the position on a printed page at which the image should be output when printed alone. It can also be used to define the image's location with respect to a larger screen or other application−specific coordinate system.

The chunk's contents are:

```
4 bytes: image position on the page, X axis (signed integer)
4 bytes: image position on the page, Y axis (signed integer)
1 byte: unit specifier
```

Both position values are signed. The following values are legal for the unit specifier:

```
0: unit is the pixel (true dimensions unspecified)
1: unit is the micrometer (also called the micron; 1/1,000,000th of a meter)
```

Conversion note: one inch is equal to exactly 25,400 micrometers.

The X position is measured rightwards from the left edge of the page to the left edge of the image; the Y position is measured downwards from the top edge of the page to the top edge of the image. Note that negative values are permitted, and denote displacement in the opposite directions. Although `oFFs` can specify an image placement which is partially or wholly outside the page boundaries, the result of such placement is application−dependent.

If present, this chunk must precede the first `IDAT` chunk.

`sCAL` Physical Scale of Image Subject

While the `pHYs` chunk is used to record the physical size of the image itself as it was scanned or as it should be printed, certain images (such as maps, photomicrographs, astronomical surveys, floor plans, and others) may benefit from knowing the actual physical dimensions of the image's subject for remote measurement and other purposes. The `sCAL` chunk serves this need. The `sCAL` chunk's contents are

```
Unit specifier: 1 byte
Pixel width:    m bytes (floating−point value as ASCII string)
Null separator: 1 byte
Pixel height:   n bytes (floating−point value as ASCII string)
```

Valid values for the unit specifier field are:

```
1   Unit is the meter
2   Unit is the radian
```

Following the unit specifier are two ASCII strings. The first string defines the physical width represented by one image pixel; the second string defines the physical height represented by one pixel. The two strings are separated by a zero byte (null character). As in the `tEXt` chunk, there is no trailing zero for the final string. Each of these strings contains a floating−point constant in the format specified above (Floating−Point Values). Both values are required to be greater than zero.

If present, this chunk must precede the first `IDAT` chunk.

GIF Conversion Chunk Descriptions

The chunks described in this section are intended specifically for preserving information found in GIF extension blocks when converting a GIF file to a PNG file. GIF−to−PNG converters should simply duplicate these blocks with the reformatting specified below. Applications that formerly made use of this information when reading a GIF file may continue to use it when reading PNG files. Since many applications ignored GIF extension blocks, it is expected that many applications will ignore these PNG chunks.

When converting single−image GIFs to PNG, these chunks should be placed in the same relative position within the PNG file as the extension block has within the GIF file. Thus converters can translate GIF extensions as they are encountered. Conversion of multiple−image GIFs is a more complex problem which will be addressed in a future document.

In a GIF file, an extension block is organized as a series of sub−blocks, where each sub−block is preceded by a length byte. PNG does not use sub−blocks, so the GIF data must be de−blocked by removal of the sub−block length bytes. In addition, the converter is expected to convert certain fields from GIF byte order (LSB first) to network byte order (MSB first), and to expand image−size−related fields from two bytes to four. Fields requiring format changes are marked in the descriptions below.

For brevity, the following chunk descriptions simply define the layout of the chunk fields and indicate the reformatting required. The semantics of the field contents remain the same as in

GIF. Readers are referred to the GIF89a specification for details. It is available on Compuserve from GRAPHSUPPORT library 16, file GIF89A.DOC, and is also widely available on Internet.

`gIFg` GIF Graphic Control Extension

The `gIFg` chunk is provided for backwards compatibility with the GIF89a Graphic Control Extension. The contents of this chunk are:

```
Disposal Method:     1 byte
User Input:          1 byte
Delay Time:          2 bytes (byte order converted from GIF)
```

Disposal Method indicates the way in which the graphic is to be treated after being displayed. User Input indicates whether user input is required before continuing. Delay Time specifies the number of hundredths (1/100) of a second to delay before continuing with the processing of the datastream. Note that this field is to be byte−order−converted.

The Transparent Color Flag and Transparency Index fields found in the GIF89a Graphic Control Extension are omitted from `gIFg`. These fields should be converted using the transparency features of basic PNG.

`gIFt` GIF Plain Text Extension

The `gIFt` chunk is provided for backwards compatibility with the GIF89a Plain Text Extension. The contents of this chunk are:

```
Text Grid Left Position:  4 bytes (byte order and size converted from GIF)
Text Grid Top Position:   4 bytes (byte order and size converted from GIF)
Text Grid Width:          4 bytes (byte order and size converted from GIF)
Text Grid Height:         4 bytes (byte order and size converted from GIF)
Character Cell Width:     1 byte
Character Cell Height:    1 byte
Text Foreground Color:    3 bytes (R,G,B samples)
Text Background Color:    3 bytes (R,G,B samples)
Plain Text Data:          n bytes
```

Text Grid Left Position, Top Position, Width, and Height specify the text area position and size in pixels. The converter must reformat these fields from 2 bytes LSB−first to 4 bytes MSB−first. Note that GIF defines the position to be relative to the upper left corner of the logical screen. If an `oFFs` chunk is also present, a decoder should assume that the `oFFs` chunk defines the offset of the image relative to the GIF logical screen; hence subtracting the `oFFs` values from the Text Grid Left and Top Positions gives the text area position relative to the main PNG image.

Character Cell Width and Height give the dimensions of each character in pixels.

Text Foreground and Background Color give the colors to be used to render text foreground and background. Note that the GIF−to−PNG converter must replace the palette index values found in the GIF Plain Text Extension block with the corresponding palette entry.

The remainder of the chunk is the text to be displayed. Note that this data is not in GIF sub−block format, but is a continuous datastream.

`gIFx` GIF Application Extension

The `gIFx` chunk is provided for backwards compatibility with the GIF89a Application Extension. The Application Extension contains application−specific information. The contents of this chunk are:

```
Application Identifier:     8 bytes
Authentication Code:        3 bytes
Application Data:           n bytes
```

The Application Identifier is a sequence of eight printable ASCII characters used to identify the application creating the Application Extension. The Authentication Code is three additional bytes which the application may use to further validate the Application Extension. The remainder of the chunk is application−specific data whose content is not defined by the GIF specification.

Note that GIF−to−PNG converters should not attempt to perform byte reordering on the contents of the Application Extension. The data is simply transcribed without any processing except for de−blocking GIF sub−blocks.

Applications that formerly used GIF Application Extensions may define special−purpose PNG chunks to replace their application extensions. If a GIF−to−PNG converter recognizes the Application Identifier and is aware of a corresponding PNG chunk, it may choose to convert the Application Extension into that PNG chunk type rather than using `gIFx`.

Chunks Not Described Here

The definitions of some public chunks are being maintained by groups other than the core PNG group. In general, these are chunks that are useful to more than one application (and thus are not private chunks), but are considered too specialized to list in the core PNG documentation.

`fRAc` Fractal Image Parameters

The `fRAc` chunk describes the parameters used to generate a fractal image. Specifications for the contents of `fRAc` chunks are maintained by Tim Wegner, `71320.675@compuserve.com`.

Text Chunk Keywords

It is expected that special−purpose keywords for `tEXt` and `zTXt` chunks will be registered and will appear in this document. However, no such keywords have yet been assigned.

End of PNG Special−Purpose Chunk Listing