pytoshop Documentation

Release 1.2.1

Michael Droettboom

Nov 30, 2018

Contents

Python library for reading and writing complex structured Photoshop files.

Contents:

CHAPTER 1

pytoshop

A Python-based library to read and write Photoshop PSD and PSB files.

Based on the specification from Adobe, but also with the help of the psd-tools source code.

- Free software: BSD license
- Documentation: https://pytoshop.readthedocs.io.

1.1 Features

- Parsing of the most important tags. This is not complete, but the infrastructure is in place to add support for more quite easily.
- Loading of complex nested layer structures, and the ability to edit them and write them back out.

CHAPTER 2

Installation

2.1 Stable release

To install pytoshop, run this command in your terminal:

```
$ pip install pytoshop
```

This is the preferred method to install pytoshop, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for pytoshop can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/mdboom/pytoshop
```

Or download the tarball:

```
$ curl  -OL https://github.com/mdboom/pytoshop/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To read a file and write it back out again:

```
import pytoshop

with open('image.psd', 'rb') as fd:
    psd = pytoshop.read(fd)

    with open('updated.psd', 'wb') as fd:
        psd.write(fd)
```

See the *API* documentation for more details.

CHAPTER 4

API

4.1 User api

read(fd)	Read a PSD file from a file-like object.
user.nested_layers	Convert a PSD file to/from nested layers.

4.1.1 pytoshop.read

pytoshop.**read**(*fd*)

Read a PSD file from a file-like object.

Parameters **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.

Returns **psdfile**

Return type *PsdFile*

4.1.2 pytoshop.user.nested_layers

Convert a PSD file to/from nested layers.

class pytoshop.user.nested_layers.**Group**(*name=''*, *visible=True*, *opacity=255*, *group_id=0*, *blend_mode=<BlendMode.pass_through: b'pass'>*, *layers=None*, *closed=True*, *metadata=None*, *layer_color=0*)

Bases: *pytoshop.user.nested_layers.Layer*

A *Layer* that may contain other *Layer* instances.

closed

Is layer closed in GUI?

layers

List of sublayers

class pytoshop.user.nested_layers.**Image**(*name=''*, *visible=True*, *opacity=255*, *group_id=0*, *blend_mode=<BlendMode.normal: b'norm'>*, *top=0*, *left=0*, *bottom=None*, *right=None*, *channels=None*, *metadata=None*, *layer_color=0*, *color_mode=None*)

Bases: *pytoshop.user.nested_layers.Layer*

A *Layer* containing image data, i.e. a leaf node.

bottom
The bottom of the layer, in pixels. If not provided, will be automatically determined from channel data.

channels
The channel image data. May be one of the following –

- A dictionary from *enums.ChannelId* to 2-D numpy arrays.
- A 3-D numpy array of the shape (num_channels, height, width)
- A list of numpy arrays where each is a channel.

It is better to use *get_channel* and *set_channel* to

color_mode
The color mode of the image.

get_channel(*color*)
Get a channel for a given color. Raises an error if the color space doesn't have the given color.

Parameters **color** (enums.ColorChannel) –

Returns **channel**

Return type 2-D numpy array

left
The left of the layer, in pixels.

right
The right of the layer, in pixels. If not provided, will be automatically determined from channel data.

set_channel(*color*, *channel*)
Get a channel for a given color. Raises an error if the color space doesn't have the given color.

Parameters

- **color** (enums.ColorChannel) –
- **channel** (*2-D numpy array*) –

top
The top of the layer, in pixels.

class pytoshop.user.nested_layers.**Layer**
Bases: object

Base class of all layers.

blend_mode
blend mode

group_id
Linked layer id

layer_color
layer color (as it appears in the layer list)

metadata

name
The name of the layer

opacity
Opacity. 0=transparent, 255=opaque

visible
Is layer visible?

pytoshop.user.nested_layers.**nested_layers_to_psd**(*layers*, *color_mode*, *version=<Version.version_1: 1>*, *compression=<Compression.rle: 1>*, *depth=None*, *size=None*, *vector_mask=False*)

Convert a hierarchy of nested *Layer* instances to a *PsdFile*.

Parameters

- **layers** (list of *Layer* instances) – The hierarchy of layers we want to create.
- **color_mode** (*enums.ColorMode*) – The color mode of the resulting PSD file (as well as the input image data).
- **version** (*enums.Version*, optional) – The version of the PSD spec to follow.
- **compression** (*enums.Compression*, optional) – The method of image compression to use for the layer image data.
- **depth** (*enums.ColorDepth*, optional) – The color depth of the resulting image. Must match the color depth of the data passed in. If not provided, the color depth will be automatically determined from the passed-in data.
- **size** (*2-tuple of int, optional*) – The shape in the form `(height, width)` of the resulting PSD file. If not provided, the height and width will be set to include all passed in layers, and the layers themselves will be adjusted so that none fall outside of the image.
- **vector_mask** (*bool, optional*) – When `True`, the mask for the layer will be a vector rectangle. This results in much smaller file sizes, but is not quite as accurately rendered by Photoshop. When `False`, a raster mask is used.

Returns **psdfile** – The resulting PSD file.

Return type *PsdFile*

pytoshop.user.nested_layers.**pprint_layers**(*layers*, *indent=0*)

Pretty-print a hierarchy of *Layer* instances.

pytoshop.user.nested_layers.**psd_to_nested_layers**(*psdfile*)

Convert a *PsdFile* instance to a hierarchy of nested *Layer* instances.

Parameters **psdfile** (`PsdFile`) – A parsed PSD file.

Returns **layers** – A representation of the parsed hierarchy from the file.

Return type list of *Layer* instances

4.2 Objects

blending_range	Manage blending ranges.
core	The core objects, including the *PsdFile* and its *Header*.
color_mode	The *ColorModeData* section.
image_data	The *ImageData* section.
image_resources	The *ImageResources* section.
layers	Sections related to image layers.
path	Handle Bézier paths.
tagged_block	*TaggedBlock* objects.

4.2.1 pytoshop.blending_range

Manage blending ranges.

class pytoshop.blending_range.**BlendingRange**(*black0=0*, *black1=0*, *white0=0*, *white1=0*, *_values=None*)

Bases: object

Blending range data.

Comprises 2 black values and 2 white values.

black0

black1

classmethod read(*fd*)

Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

white0

white1

write(*fd*, *header*)

Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.blending_range.**BlendingRangePair**(*src=None*, *dst=None*)

Bases: object

Blending range pair.

The combination of a source and destination blending range.

dst

length(*header*)

The length of the section, in bytes, not including its header.

classmethod read(*fd*)

Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

src

total_length (*header*)
The length of the section, in bytes, including its header.

write (*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.blending_range.**BlendingRanges** (*composite_gray_blend=None*, *channels=None*)

Bases: object

All of the layer blending range data.

Consists of a composite gray blend pair followed by N additional pairs.

channels
List of additional *BlendingRangePair* instances.

composite_gray_blend
Composite gray *BlendingRangePair*.

length (*header*)
The length of the section, in bytes, not including its header.

classmethod read (*fd*, *num_channels*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

total_length (*header*)
The length of the section, in bytes, including its header.

write (*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

4.2.2 pytoshop.core

The core objects, including the *PsdFile* and its *Header*.

class pytoshop.core.**Header** (*version=<Version.version_1: 1>*, *num_channels=1*, *height=1*, *width=1*, *depth=<ColorDepth.depth8: 8>*, *color_mode=<ColorMode.rgb: 3>*)

Bases: object

Manages the header at the start of a PSD/PSB file.

color_mode
Color mode of the file. See *enums.ColorMode*.

depth
Number of bits per channel. See *enums.ColorDepth*.

classmethod header_read(*fd*)
Instantiate from a file-like object.

Parameters **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.

height
Height of the image (in pixels).

max_size_mapping = {1: 30000, 2: 300000}

num_channels
Number of color channels in the file.

shape

version
The version of the file format. See *enums.Version*.

width
Width of the image (in pixels).

write(*fd*)
Write to a file-like object.

Parameters **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.

class pytoshop.core.**PsdFile**(*version=<Version.version_1: 1>*, *num_channels=1*, *height=1*, *width=1*, *depth=<ColorDepth.depth8: 8>*, *color_mode=<ColorMode.rgb: 3>*, *color_mode_data=None*, *image_resources=None*, *layer_and_mask_info=None*, *image_data=None*, *compression=<Compression.raw: 0>*)
Bases: *pytoshop.core.Header*

Represents an entire PSD file.

color_mode_data
Color mode data section. See *color_mode.ColorModeData*.

image_data
Image data. See *image_data.ImageData*.

image_resources
Image resources. See *image_resources.ImageResources*.

layer_and_mask_info
Image resources. See *image_resources.ImageResources*.

classmethod read(*fd*)
Instantiate from a file-like object.

Parameters **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.

write(*fd*)
Write to a file-like object.

Parameters **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.

4.2.3 pytoshop.color_mode

The *ColorModeData* section.

class pytoshop.color_mode.**ColorModeData**(*data=b''*)
Bases: object

Color mode data section.

Only indexed color and duotone (see *core.Header.color_mode*) have color mode data.

Indexed color images: length is 768; color data contains the color table for the image, in non-interleaved order.

Duotone images: color data contains the duotone specification (the format of which is not documented). Other applications that read Photoshop files can treat a duotone image as a gray image, and just preserve the contents of the duotone information when reading and writing the file.

Note that pytoshop doesn't do anything meaningful for color mode data, and only stores the raw bytes in order to round-trip.

data

length(*header*)
The length of the section, in bytes, not including its header.

classmethod read(*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

write(*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

4.2.4 pytoshop.image_data

The *ImageData* section.

class pytoshop.image_data.**ImageData**(*channels=None*, *fd=None*, *offset=None*, *size=None*, *height=None*, *width=None*, *num_channels=None*, *depth=None*, *version=None*, *compression=<Compression.raw: 0>*)
Bases: object

Stores (non-layer) image data.

channels

compression
Type of compression. See *enums.Compression*.

classmethod read(*fd*, *header*)

shape

write(*fd*, *header*)

4.2.5 pytoshop.image_resources

The *ImageResources* section.

Image resource blocks are the basic building unit of several file formats, including Photoshop's native file format, JPEG, and TIFF. Image resources are used to store non-pixel data associated with images, such as pen tool paths.

class pytoshop.image_resources.**AlphaIdentifiers**(*name=''*, *identifiers=[]*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data_length(*header*)

identifiers
Alpha indentifiers

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**BackgroundColor**(*name=''*, *color_space=<ColorSpace.rgb: 0>*, *color=[]*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

Background color.

color
The color data. If the color data does not require 4 values, the extra values are undefined and should be included as zeros.

color_space
The color space. See *enums.ColorSpace*

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**BorderInfo**(*name=''*, *border_width_num=0*, *border_width_den=1*, *unit=<Units.inches: 1>*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

Border information.

border_width_den
Border width denominator

border_width_num
Border width numerator

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

unit
Unit. See *enums.Units*.

write_data(*fd*, *header*)

class pytoshop.image_resources.**CopyrightFlag**(*name=''*, *copyright=False*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

copyright
Is copyrighted?

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**DocumentSpecificIdsSeedNumber**(*name=''*, *base_value=0*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

base_value
Base value

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**EffectsVisible**(*name=''*, *visible=False*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

visible
Are effects visible?

write_data(*fd*, *header*)

class pytoshop.image_resources.**GenericImageResourceBlock**(*name=''*, *resource_id=0*, *data=b''*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data
Raw data of image resource.

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

resource_id
Type of image resource.

write_data(*fd*, *header*)

class pytoshop.image_resources.**GlobalAltitude**(*name=''*, *altitude=0*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

altitude
Global altitude

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**GlobalAngle**(*name=''*, *angle=0*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

angle
Global light angle for the effect layer

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**GridAndGuidesInfo**(*name=''*, *grid_hori=0*, *grid_vert=0*, *guides=[]*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

Grid and guides resource.

data_length(*header*)

grid_hori
Document-specific grid (horizontal). In 1/32 pt.

grid_vert
Document-specific grid (vertical). In 1/32 pt.

guides
Guides. See *GuideResourceBlock*.

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

version

write_data(*fd*, *header*)

class pytoshop.image_resources.**GuideResourceBlock**(*location=0*, *direction=<GuideDirection.vertical: 0>*)
Bases: object

data_length(*header*)

direction
Guide direction. See *enums.GuideDirection*.

location
Location of guide in document coordinates.

classmethod read(*fd*, *header*)

write(*fd*, *header*)

class pytoshop.image_resources.**ImageResourceBlock**
Bases: object

Stores a single image resource block.

pytoshop currently doesn't deeply parse image resource blocks. The raw data is merely retained for round-tripping.

data_length(*header*)

length(*header*)
The length of the section, in bytes, not including its header.

name
Name of image resource.

classmethod read(*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

classmethod read_data (*fd*, *resource_id*, *name*, *data_length*, *header*)

resource_id
Type of image resource.

total_length (*header*)
The length of the section, in bytes, including its header.

write (*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

write_data (*fd*, *header*)

class pytoshop.image_resources.**ImageResourceUnicodeString** (*name=''*, *value=''*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data_length (*header*)

classmethod read_data (*fd*, *resource_id*, *name*, *length*, *header*)

value

write_data (*fd*, *header*)

class pytoshop.image_resources.**ImageResources** (*blocks=[]*)
Bases: object

The image resource block section.

blocks
List of all *ImageResourceBlock* items.

get_block (*resource_id*)
Get the first block with the given resource id.

length (*header*)
The length of the section, in bytes, not including its header.

classmethod read (*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

total_length (*header*)
The length of the section, in bytes, including its header.

write (*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.

- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.image_resources.**LayersGroupInfo**(*name=''*, *group_ids=[]*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

Layers group information.

Indicates which layers are locked together.

data_length(*header*)

group_ids

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.image_resources.**PrintFlags**(*name=''*, *labels=False*, *crop_marks=False*, *color_bars=False*, *registration_marks=False*, *negative=False*, *flip=False*, *interpolate=False*, *caption=False*, *print_flags=False*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

Print flags.

caption

color_bars
color bars

crop_marks
crop marks

data_length(*header*)

flip

interpolate

labels

negative

print_flags
print flags

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

registration_marks
registration marks

write_data(*fd*, *header*)

class pytoshop.image_resources.**PrintScale**(*name=''*, *style=<PrintScaleStyle.centered: 0>*, *x=0.0*, *y=0.0*, *scale=0.0*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

scale

style
Style. See *enums.PrintScaleStyle*.

write_data(*fd*, *header*)

x
x location

y
y location

class pytoshop.image_resources.**UnicodeAlphaNames**(*name=''*, *value=''*)
Bases: *pytoshop.image_resources.ImageResourceUnicodeString*

class pytoshop.image_resources.**Url**(*name=''*, *url=b''*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data_length(*header*)

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

url
URL

write_data(*fd*, *header*)

class pytoshop.image_resources.**VersionInfo**(*name=''*, *version=0*, *has_real_merged_data=False*, *writer=''*, *reader=''*, *file_version=0*)
Bases: *pytoshop.image_resources.ImageResourceBlock*

data_length(*header*)

file_version
file version

has_real_merged_data
has real merged data?

classmethod read_data(*fd*, *resource_id*, *name*, *length*, *header*)

reader
reader name

version

write_data(*fd*, *header*)

writer
writer name

class pytoshop.image_resources.**WorkflowUrl**(*name=''*, *value=''*)
Bases: *pytoshop.image_resources.ImageResourceUnicodeString*

4.2.6 pytoshop.layers

Sections related to image layers.

class pytoshop.layers.**ChannelImageData**(*image=None*, *fd=None*, *offset=None*, *size=None*, *shape=None*, *depth=None*, *version=None*, *compression=<Compression.raw: 0>*)
Bases: object

A single plane of channel image data.

compression
Compression method. See *enums.Compression*.

dtype

image

classmethod read(*fd*, *header*, *shape*, *size*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

shape

write(*fd*, *header*, *shape*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.layers.**GlobalLayerMaskInfo**(*overlay_color_space=b'x00x00x00x00x00x00x00x00x00x00'*, *opacity=100*, *kind=<LayerMaskKind.use_value_stored_per_layer: 128>*)

Bases: object

Global layer mask info.

kind
Layer mask kind. See *enums.LayerMaskKind*

opacity
Opacity. 0=transparent, 100=opaque

overlay_color_space
Undocumented

classmethod read(*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

write(*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.layers.**LayerAndMaskInfo**(*layer_info=None*, *global_layer_mask_info=None*, *additional_layer_info=None*)

Bases: object

Layer and mask information section.

additional_layer_info
List of additional layer info. See *TaggedBlock*.

additional_layer_info_map
A mapping from tagged block codes to *tagged_block.TaggedBlock* instances.

This is a convenience to more easily get associated tagged blocks.

global_layer_mask_info
Global layer mask info. See *GlobalLayerMaskInfo*.

layer_info
Layer info. See *LayerInfo*.

classmethod read(*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

write(*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.layers.**LayerInfo**(*layer_records=None*, *use_alpha_channel=False*)
Bases: object

A set of *LayerRecord* instances.

layer_records
List of *LayerRecord* instances

classmethod read(*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

use_alpha_channel
Indicates that the first channel contains transparency data for the merged result.

write(*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.layers.**LayerMask**(*top=0*, *left=0*, *bottom=0*, *right=0*, *default_color=False*, *position_relative_to_layer=False*, *layer_mask_disabled=False*, *invert_layer_mask_when_blending=False*, *user_mask_from_rendering_other_data=False*, *user_mask_density=None*, *user_mask_feather=None*, *vector_mask_density=None*, *vector_mask_feather=None*, *real_flags=0*, *real_user_mask_background=False*, *real_top=0*, *real_left=0*, *real_bottom=0*, *real_right=0*)

Bases: object

Layer mask / adjustment layer data.

bottom
Bottom of rectangle enclosing layer mask

default_color
Default color for mask

height
Height of the mask layer.

invert_layer_mask_when_blending
Invert layer mask when blending (obsolete)

layer_mask_disabled
Layer mask disabled

left
Left of rectangle enclosing layer mask

length (*header*)
The length of the section, in bytes, not including its header.

position_relative_to_layer
position relative to layer

classmethod read (*fd*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

real_bottom

real_flags

real_height
Real height of the mask layer.

real_left

real_right

real_shape
Real shape of the mask layer (height, width).

real_top

real_user_mask_background

real_width
Real width of the mask layer.

right
Right of rectangle enclosing layer mask

shape
Shape of the mask layer (height, width).

top
Top of rectangle enclosing layer mask

total_length (*header*)
The length of the section, in bytes, including its header.

user_mask_density
User mask density

user_mask_feather
User mask feather

user_mask_from_rendering_other_data
Indicates that the user mask actually came from rendering other data.

vector_mask_density
Vector mask density

vector_mask_feather
Vector mask feather

width
Width of the mask layer.

write (*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

class pytoshop.layers.**LayerRecord**(*top=0*, *left=0*, *bottom=0*, *right=0*, *blend_mode=<BlendMode.normal: b'norm'>*, *opacity=255*, *clipping=False*, *transparency_protected=False*, *visible=True*, *pixel_data_irrelevant=False*, *name=''*, *channels=None*, *blocks=None*, *color_mode=None*)

Bases: object

Layer record.

There is one of these per logical layer in the file.

blend_mode
Blend mode. See *enums.BlendMode*

blending_ranges

blocks
List of *tagged_block.TaggedBlock* items with additional information about this layer.

blocks_map
A mapping from tagged block codes to *tagged_block.TaggedBlock* instances.

This is a convenience to more easily get associated tagged blocks.

bottom
Bottom of rectangle enclosing layer

channels
Dictionary from *enums.ChannelId* to *ChannelImageData*.

For safety against different color modes, it is better to use *get_channel* and *set_channel*.

clipping
Clipping. False=base, True=non-base

get_channel(*color*)
Get a channel for a given color. Raises an error if the color space doesn't have the given color.

Parameters **color** (enums.ColorChannel) –

Returns channel

Return type *ChannelImageData*

height
Height of the layer.

left
Left of rectangle enclosing layer

mask

name
Name of layer

opacity
Opacity. 0=transparent, 255=opaque

pixel_data_irrelevant
Pixel data is irrelevant to appearance of document

classmethod read(*fd*, *header*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

read_channel_data(*fd*, *header*)
Read the *ChannelImageData* for this layer.

right
Right of rectangle enclosing layer

set_channel(*color*, *channel*)
Set a channel for a given color. Raises an error if the color space doesn't have the given color.

Parameters

- **color** (enums.ColorChannel) –
- **channel** (ChannelImageData) –

shape
Shape of the layer `(height, width)`.

top
Top of rectangle enclosing layer

transparency_protected
Transparency protected

visible
Visible

width
Width of the layer.

write (*fd*, *header*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

write_channel_data (*fd*, *header*)
Write the *ChannelImageData* for this layer.

4.2.7 pytoshop.path

Handle Bézier paths.

class pytoshop.path.**ClipboardRecord**(*top=0.0*, *left=0.0*, *bottom=0.0*, *right=0.0*, *resolution=0*)
Bases: *pytoshop.path.PathRecord*

bottom
bottom, in pixels

left
left, in pixels

classmethod read_data(*fd*, *header*)

resolution

right
right, in pixels

top
top, in pixels

write_data(*fd*, *header*)

class pytoshop.path.**ClosedSubpathBezierKnotLinked**(*y0=0.0*, *x0=0.0*, *y1=None*, *x1=None*, *y2=None*, *x2=None*)
Bases: pytoshop.path._PointRecord

class pytoshop.path.**ClosedSubpathBezierKnotUnlinked**(*y0=0.0*, *x0=0.0*, *y1=None*, *x1=None*, *y2=None*, *x2=None*)
Bases: pytoshop.path._PointRecord

class pytoshop.path.**ClosedSubpathLengthRecord**(*num_knots=0*)
Bases: pytoshop.path._LengthRecord

class pytoshop.path.**InitialFillRuleRecord**(*all_pixels=False*)
Bases: *pytoshop.path.PathRecord*

all_pixels
Fill starts with all pixels

classmethod read_data(*fd*, *header*)

write_data(*fd*, *header*)

class pytoshop.path.**OpenSubpathBezierKnotLinked**(*y0=0.0*, *x0=0.0*, *y1=None*, *x1=None*, *y2=None*, *x2=None*)
Bases: pytoshop.path._PointRecord

class pytoshop.path.**OpenSubpathBezierKnotUnlinked**(*y0=0.0*, *x0=0.0*, *y1=None*, *x1=None*, *y2=None*, *x2=None*)
Bases: pytoshop.path._PointRecord

class pytoshop.path.**OpenSubpathLengthRecord**(*num_knots=0*)
Bases: pytoshop.path._LengthRecord

class pytoshop.path.**PathFillRuleRecord**
Bases: *pytoshop.path.PathRecord*

classmethod read_data(*fd*, *header*)

write_data(*fd*, *header*)

class pytoshop.path.**PathRecord**
Bases: object

length(*header*)

classmethod read(*fd*, *header*)

classmethod read_data(*fd*, *header*)

type

write(*fd*, *header*)

write_data(*fd*, *header*)

class pytoshop.path.**PathResource**(*path_records=[]*)
Bases: object

classmethod from_rect(*top*, *left*, *bottom*, *right*, *all_pixels=True*)

length(*header*)
The length of the section, in bytes, not including its header.

path_records
List of *PathRecord* instances.

classmethod read(*fd*, *length*, *header*)

write(*fd*, *header*)

4.2.8 pytoshop.tagged_block

TaggedBlock objects.

class pytoshop.tagged_block.**GenericTaggedBlock**(*code=b”*, *data=b”*)
Bases: *pytoshop.tagged_block.TaggedBlock*

A generic *TaggedBlock* subclass for tag codes pytoshop doesn't know about.

code

data

data_length(*header*)

classmethod read_data(*fd*, *code*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.tagged_block.**LayerColor**(*color=0*)
Bases: *pytoshop.tagged_block.TaggedBlock*

color
Color

data_length(*header*)

classmethod read_data(*fd*, *code*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.tagged_block.**LayerId**(*id=0*)
Bases: *pytoshop.tagged_block.TaggedBlock*

data_length(*header*)

id
Layer id

classmethod read_data(*fd*, *code*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.tagged_block.**LayerNameSource**(*id=0*)
Bases: *pytoshop.tagged_block.TaggedBlock*

data_length(*header*)

id
The layer id of the source of the name of this layer

classmethod read_data(*fd*, *code*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.tagged_block.**MetadataSetting**(*datas=None*)
Bases: *pytoshop.tagged_block.TaggedBlock*

data_length(*header*)

datas

classmethod read_data(*fd*, *code*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.tagged_block.**NestedSectionDividerSetting**(*type=<SectionDividerSetting.open: 1>*, *key=None*, *subtype=None*)
Bases: pytoshop.tagged_block._SectionDividerSetting

class pytoshop.tagged_block.**SectionDividerSetting**(*type=<SectionDividerSetting.open: 1>*, *key=None*, *subtype=None*)
Bases: pytoshop.tagged_block._SectionDividerSetting

class pytoshop.tagged_block.**TaggedBlock**
Bases: object

code

data_length(*header*)

static is_long_length(*code*, *header*)

length(*header*)
The length of the section, in bytes, not including its header.

classmethod read(*fd*, *header*, *padding=1*)
Instantiate from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be readable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

classmethod read_data(*fd*, *code*, *length*, *header*)

total_length(*header*, *padding=1*)
The length of the section, in bytes, including its header.

write(*fd*, *header*, *padding=1*)
Write to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be writable, seekable and open in binary mode.
- **header** (*PsdFile object*) – An object to get global file information from.

write_data(*fd*, *header*)

class pytoshop.tagged_block.**UnicodeLayerName**(*name=''*)
Bases: *pytoshop.tagged_block.TaggedBlock*

data_length(*header*)

name
The name of the layer.

classmethod read_data(*fd*, *code*, *length*, *header*)

write_data(*fd*, *header*)

class pytoshop.tagged_block.**VectorMask**(*version=3*, *invert=False*, *not_link=False*, *disable=False*, *path_resource=None*)
Bases: *pytoshop.tagged_block.TaggedBlock*

data_length(*header*)

disable
Disable mask

invert
Invert mask

not_link
Don't link mask

path_resource
path.PathResource instance'

classmethod read_data(*fd*, *code*, *length*, *header*)

version
Vector mask block version

write_data(*fd*, *header*)

4.3 Enums

enums	Enumerated values used throughout the library.

4.3.1 pytoshop.enums

Enumerated values used throughout the library.

class pytoshop.enums.**BlendMode**

Bases: bytes, enum.Enum

Layer blend mode.

color = b'colr'

color_burn = b'idiv'

color_dodge = b'div '

darken = b'dark'

darker_color = b'dkCl'

difference = b'diff'

dissolve = b'diss'

divide = b'fdiv'

exclusion = b'smud'

hard_light = b'hLit'

hard_mix = b'hMix'

hue = b'hue '

lighten = b'lite'

lighter_color = b'lgCl'

linear_burn = b'lbrn'

linear_dodge = b'lddg'

linear_light = b'lLit'

luminosity = b'lum '

multiply = b'mul '

normal = b'norm'

overlay = b'over'

pass_through = b'pass'

pin_light = b'pLit'

saturation = b'sat '

screen = b'scrn'

soft_light = b'sLit'

subtract = b'fsub'

vivid_light = b'vLit'

class pytoshop.enums.**ChannelId**

Bases: enum.IntEnum

Channel id.

Used to map channel data to image planes in layers.

The meaning of the positive numbers depends on the *ColorMode* in effect.

L = 0

a = 1

b = 2

bitmap = 0

black = 3

blue = 2

cyan = 0

default = 0

gray = 0

green = 1

magenta = 1

real_user_layer_mask = -3

red = 0

transparency = -1

user_layer_mask = -2

yellow = 2

class pytoshop.enums.**ColorChannel**

Bases: enum.IntEnum

Color channel names, with unique values so we can check that they apply to the expected color mode.

L = 10

a = 11

b = 12

bitmap = 0

black = 9

blue = 5

cyan = 6

default = 2

gray = 1

green = 4

magenta = 7

real_user_layer_mask = -3

red = 3

transparency = -1

user_layer_mask = -2

yellow = 8

class pytoshop.enums.**ColorDepth**
Bases: enum.IntEnum

Color depth (bits-per-pixel-per-channel).

Supported values are 1, 8, 16, and 32.

depth1 = 1

depth16 = 16

depth32 = 32

depth8 = 8

class pytoshop.enums.**ColorMode**
Bases: enum.IntEnum

Color mode.

bitmap = 0

cmyk = 4

duotone = 8

grayscale = 1

indexed = 2

lab = 9

multichannel = 7

rgb = 3

class pytoshop.enums.**ColorSpace**
Bases: enum.IntEnum

Color space ids.

Defines the meaning of the 4 color values in a color structure.

- `rgb`: The first three values are *red*, *green*, and *blue*. The are full 16-bit unsigned values as in Apple's `RGBColor` data structure.
- `hsb`: The first three values are *hue*, *saturation*, and *brightness*. The are full unsigned 16-bit values as in Apple's `HSVColor` data structure.
- `cmyk`: The four values are *cyan*, *magenta*, *yellow* and *black*. The are full unsigned 16-bit values. 0 = 100% ink.
- `lab`: The first three values are *lightness*, *a chrominance*, and *b chrominance*. Lightness is a 16-bit value from 0 to 10000. Chrominance components are each 16-bit values from -12800 to 12700. Gray values are represented by chrominance components of 0.
- `grayscale`: The first value in the color data is the gray value from 0 to 10000.

Additional values are for "custom color spaces" which are not as well documented:

- `pantone`: Pantone® matching system.
- `focoltone`: Focoltone matching system.
- `trumatch`: Trumatch color.
- `toyo_88_colorfinder_1050`: Toyo 88 colorfinder 1050.

- `hks`: HKS colors.

cmyk = 2

focoltone = 4

grayscale = 8

hks = 10

hsb = 1

lab = 7

pantone = 3

rgb = 0

toyo_88_colorfinder_1050 = 6

trumatch = 5

class pytoshop.enums.**Compression**
Bases: enum.IntEnum

Compression mode.

- `raw`: raw image data.
- `rle`: Run length encoded (RLE) compressed. The RLE compression is the same compression algorithm used by the Macintosh ROM routine PackBits, and the TIFF standard.
- `zip`: Zip (zlib) without prediction.
- `zip_prediction`: Zip (zlib) with prediction.

raw = 0

rle = 1

zip = 2

zip_prediction = 3

class pytoshop.enums.**GuideDirection**
Bases: enum.IntEnum

An enumeration.

horizontal = 1

vertical = 0

class pytoshop.enums.**ImageResourceID**
Bases: enum.IntEnum

Ids for Image Resource blocks.

alpha_channel_names = 1006

alpha_identifiers = 1053

alternate_duotone_colors = 1066

alternate_spot_colors = 1067

auto_save_file_path = 1086

auto_save_format = 1087

```
background_color = 1010
border_info = 1009
caption = 1008
caption_digest = 1061
color_halftoning_info = 1013
color_samplers_resource = 1073
color_transfer_funcs = 1016
copyright_flag = 1034
count_info = 1080
display_info = 1077
document_specific_ids_seed_number = 1044
duotone_halftoning_info = 1014
duotone_image_info = 1018
duotone_transfer_funcs = 1017
effective_black_and_white = 1019
effects_visible = 1042
eps_options = 1021
exif_data_1 = 1058
exif_data_2 = 1059
global_altitude = 1049
global_angle = 1037
grayscale_and_multichannel_halftoning_info = 1012
grayscale_and_multichannel_transfer_func = 1015
grid_and_guides_info = 1032
hdr_toning_info = 1070
icc_profile = 1039
icc_untagged_profile = 1041
image_mode_for_raw = 1029
image_ready_7_rollover_expanded_state = 7003
image_ready_data_sets = 7001
image_ready_default_selected_state = 7002
image_ready_rollover_expanded_state = 7004
image_ready_save_layer_settings = 7005
image_ready_variables = 7000
image_ready_version = 7006
indexed_color_table_count = 1046
```

```
iptc_naa = 1028
jpeg_quality = 1030
jump_to_xpep = 1052
layer_comps = 1065
layer_groups_enabled = 1072
layer_selection_ids = 1069
layer_state_info = 1024
layers_group_info = 1026
lightroom_workflow = 8000
mac_nsprintinfo = 1084
mac_page_format_info = 1002
mac_print_info = 1001
measurement_scale = 1074
name_of_clipping_path = 2999
onion_skins = 1078
origin_path_info = 3000
path_selection_state = 1088
pixel_aspect_ratio = 1064
print_flags = 1011
print_flags_info = 10000
print_info = 1071
print_info_cs5 = 1082
print_scale = 1062
print_style = 1083
quick_mask_info = 1022
resolution_info = 1005
sheet_disclosure = 1076
slices = 1050
spot_halftone = 1043
thumbnail_resource = 1036
timeline_info = 1075
transparency_index = 1047
unicode_alpha_names = 1045
url = 1035
url_list = 1054
version_info = 1057
```

watermark = 1040

win_devmode = 1085

workflow_url = 1051

xmp_metadata = 1060

class pytoshop.enums.**LayerMaskKind**
Bases: enum.IntEnum

Layer mask kind.

According to the spec, only use_value_stored_per_layer is preferred. The other are retained for backward compatibility only.

color_protected = 1

color_selected = 0

use_value_stored_per_layer = 128

class pytoshop.enums.**PathRecordType**
Bases: enum.IntEnum

An enumeration.

clipboard_record = 7

closed_subpath_bezier_knot_linked = 1

closed_subpath_bezier_knot_unlinked = 2

closed_subpath_length = 0

initial_fill_rule_record = 8

open_subpath_bezier_knot_linked = 4

open_subpath_bezier_knot_unlinked = 5

open_subpath_length = 3

path_fill_rule_record = 6

class pytoshop.enums.**PrintScaleStyle**
Bases: enum.IntEnum

An enumeration.

centered = 0

size_to_fit = 1

user_defined = 2

class pytoshop.enums.**SectionDividerSetting**
Bases: enum.IntEnum

Section divider setting.

Used for the mode of grouped layers.

- any_other: any other type of layer
- open: Display an open folder
- closed: Display a closed folder
- bounding: Bounding section divider, hidden in GUI.

any_other = 0

bounding = 3

closed = 2

open = 1

class pytoshop.enums.**Units**
Bases: enum.IntEnum

An enumeration.

cm = 2

columns = 5

inches = 1

picas = 4

points = 3

class pytoshop.enums.**Version**
Bases: enum.IntEnum

The PSD file version.

Version 1 is the classic "PSD" file.

Version 2 is the large document format "PSB" file which supports documents up to 300,000 pixels in any dimension.

psb = 2

psd = 1

version_1 = 1

version_2 = 2

4.4 Codecs

codecs	Coders and decoders for the various compression types in PSD.

4.4.1 pytoshop.codecs

Coders and decoders for the various compression types in PSD.

pytoshop.codecs.**compress_constant_raw**(*fd*, *value*, *width*, *rows*, *depth*, *version*)
Write a virtual image containing a constant to a raw stream.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **value** (*int*) – The constant value in the generated virtual image.
- **width** (*int*) – The width of the image, in pixels.
- **rows** (*int*) – The number of rows in the image, in pixels. This is `height * num_channels`.

- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

pytoshop.codecs.**compress_constant_rle**(*fd*, *value*, *width*, *rows*, *depth*, *version*)
Write a virtual image containing a constant to a runlength-encoded stream.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **value** (*int*) – The constant value in the generated virtual image.
- **width** (*int*) – The width of the image, in pixels.
- **rows** (*int*) – The number of rows in the image, in pixels. This is `height * num_channels`.
- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

pytoshop.codecs.**compress_constant_zip**(*fd*, *value*, *width*, *rows*, *depth*, *version*)
Write a virtual image containing a constant to a zip compressed stream.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **value** (*int*) – The constant value in the generated virtual image.
- **width** (*int*) – The width of the image, in pixels.
- **rows** (*int*) – The number of rows in the image, in pixels. This is `height * num_channels`.
- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

pytoshop.codecs.**compress_constant_zip_prediction**(*fd*, *value*, *width*, *rows*, *depth*, *version*)
Write a virtual image containing a constant to a zip with prediction compressed stream.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **value** (*int*) – The constant value in the generated virtual image.
- **width** (*int*) – The width of the image, in pixels.
- **rows** (*int*) – The number of rows in the image, in pixels. This is `height * num_channels`.
- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

pytoshop.codecs.**compress_image**(*fd*, *image*, *compression*, *shape*, *num_channels*, *depth*, *version*)
Write an image with the given compression type.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **image** (*2-D numpy array or scalar*) – The image to compress. Must be unsigned integer with 8, 16 or 32 bits. If *depth* is 1, the array should have dtype `uint8` with a byte per pixel. If a scalar, a "virtual" image will be used as if the image contained only that constant value.
- **compression** (`enums.Compression`) – The compression format to use. See *enums.Compression*.
- **shape** (*2-tuple of int*) – The shape of the image `(height, width)`. If *image* is an array, the *shape* is used to confirm it has the correct shape. If *image* is a constant, *shape* is used to generate the virtual constant image.
- **num_channels** (*int*) – The number of color channels in the image. If *image* is an array, the *num_channels* is used to confirm it has the correct number of channels. If *image* is a constant, *num_channels* is used to generate the virtual constant image.
- **depth** (`enums.ColorDepth`) – The bit depth of the image. See *enums.ColorDepth*. If *image* is an array, the *depth* is used to confirm it has the correct number of channels. If *image* is a constant, *depth* is used to generate the virtual constant image.
- **version** (`enums.Version`) – The version of the PSD file. See *enums.Version*.

`pytoshop.codecs.`**compress_raw**(*fd*, *image*, *depth*, *version*)

Write a Numpy array to raw bytes in a file.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **image** (*2-D numpy array*) – The image to compress. Must be unsigned integer with 8, 16 or 32 bits. If *depth* is 1, the array should have dtype `uint8` with a byte per pixel.
- **depth** (`enums.ColorDepth`) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (`enums.Version`) – The version of the PSD file. See *enums.Version*.

`pytoshop.codecs.`**compress_rle**(*fd*, *image*, *depth*, *version*)

Write a Numpy array to a run length encoded stream.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **image** (*2-D numpy array*) – The image to compress. Must be unsigned integer with 8, 16 or 32 bits. If *depth* is 1, the array should have dtype `uint8` with a byte per pixel.
- **depth** (`enums.ColorDepth`) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (`enums.Version`) – The version of the PSD file. See *enums.Version*.

`pytoshop.codecs.`**compress_zip**(*fd*, *image*, *depth*, *version*)

Write a Numpy array to a zip (zlib) compressed stream.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **image** (*2-D numpy array*) – The image to compress. Must be unsigned integer with 8, 16 or 32 bits. If *depth* is 1, the array should have dtype `uint8` with a byte per pixel.

- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

pytoshop.codecs.**compress_zip_prediction**(*fd*, *image*, *depth*, *version*)
Write a Numpy array to a zip (zlib) with prediction compressed stream.

Not supported for 1- or 32-bit images.

Parameters

- **fd** (*file-like object*) – Writable file-like object, open in binary mode.
- **image** (*2-D numpy array*) – The image to compress. Must be unsigned integer with 8, 16 or 32 bits. If *depth* is 1, the array should have dtype `uint8` with a byte per pixel.
- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

pytoshop.codecs.**decompress_image**(*data*, *compression*, *shape*, *depth*, *version*)
Decompress data with the given compression.

Parameters

- **data** (*bytes*) – The raw bytes from the file.
- **compression** (enums.Compression) – The compression format to use. See *enums.Compression*.
- **shape** (*2-tuple of int*) – The shape of the resulting array, `(height, width)`.
- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

Returns **image** – The image data as a Numpy array.

Return type numpy array

pytoshop.codecs.**decompress_raw**(*data*, *shape*, *depth*, *version*)
Converts raw data to a Numpy array.

Parameters

- **data** (*bytes*) – The raw bytes from the file.
- **shape** (*2-tuple of int*) – The shape of the resulting array, `(height, width)`.
- **depth** (enums.ColorDepth) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (enums.Version) – The version of the PSD file. See *enums.Version*.

Returns **image** – The image data as a Numpy array. If *depth* is 1, the array is expanded to `uint8` with a byte per pixel.

Return type numpy array

pytoshop.codecs.**decompress_rle**(*data*, *shape*, *depth*, *version*)
Decompress run length encoded data.

Parameters

- **data** (*bytes*) – The raw bytes from the file.

- **shape** (*2-tuple of int*) – The shape of the resulting array, `(height, width)`.
- **depth** (`enums.ColorDepth`) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (`enums.Version`) – The version of the PSD file. See *enums.Version*.

Returns **image** – The image data as a Numpy array. If *depth* is 1, the array is expanded to `uint8` with a byte per pixel.

Return type numpy array

`pytoshop.codecs.`**decompress_zip**(*data*, *shape*, *depth*, *version*)

Decompress zip (zlib) encoded data.

Parameters

- **data** (*bytes*) – The raw bytes from the file.
- **shape** (*2-tuple of int*) – The shape of the resulting array, `(height, width)`.
- **depth** (`enums.ColorDepth`) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (`enums.Version`) – The version of the PSD file. See *enums.Version*.

Returns **image** – The image data as a Numpy array. If *depth* is 1, the array is expanded to `uint8` with a byte per pixel.

Return type numpy array

`pytoshop.codecs.`**decompress_zip_prediction**(*data*, *shape*, *depth*, *version*)

Decompress zip (zlib) with prediction encoded data.

Not supported for 1- or 32-bit images.

Parameters

- **data** (*bytes*) – The raw bytes from the file.
- **shape** (*2-tuple of int*) – The shape of the resulting array, `(height, width)`.
- **depth** (`enums.ColorDepth`) – The bit depth of the image. See *enums.ColorDepth*.
- **version** (`enums.Version`) – The version of the PSD file. See *enums.Version*.

Returns **image** – The image data as a Numpy array. If *depth* is 1, the array is expanded to `uint8` with a byte per pixel.

Return type numpy array

`pytoshop.codecs.`**normalize_image**(*image*, *depth*)

4.5 Utilities

util	Miscellaneous utilities.

4.5.1 pytoshop.util

Miscellaneous utilities.

`pytoshop.util.`**assert_is_list_of**(*value*, *cls*, *min=None*, *max=None*)

If value is not a list of cls instances, raises TypeError.

pytoshop.util.**decode_unicode_string**(*data*)
Decode Photoshop's definition of a Unicode String.

pytoshop.util.**do_byteswap**(*arr*)
Return a copy of an array, byteswapped.

pytoshop.util.**encode_unicode_string**(*s*)
Encode Photoshop's definition of a Unicode String.

pytoshop.util.**ensure_bigendian**(*arr*)
Ensure that a Numpy array is in big-endian order.

Returns a copy if the endianness needed to be changed.

pytoshop.util.**ensure_native_endian**(*arr*)
Ensure that a Numpy array is in native-endian order.

Returns a copy if the endianness needed to be changed.

pytoshop.util.**get_channel**(*color*, *color_mode*, *channels*)

pytoshop.util.**log**(*msg*, **args*)
Print a logging message if debugging is turned on.

pytoshop.util.**needs_byteswap**(*arr*)
Returns True if the array needs to be byteswapped.

pytoshop.util.**pack_bitflags**(**values*)
Pack separate booleans back into a bit field.

pytoshop.util.**pad**(*number*, *divisor*)
Pads an integer up to the given divisor.

pytoshop.util.**pascal_string_length**(*value*, *padding=1*)
Calculates the total length of writing a UTF-8-encoded Pascal string to disk.

Parameters **value** (*str*) – A unicode string value.

Returns **length** – The length, in bytes.

Return type int

pytoshop.util.**read_pascal_string**(*fd*, *padding=1*)
Read a UTF-8-encoded Pascal string from a file.

Parameters

- **fd** (*file-like object*) – Must be opened for reading, seekable and in binary mode.
- **padding** (*int, optional*) – If provided, additional pad bytes will be read until the total amount read is a multiple of padding.

Returns **value** – The unicode value of the string.

Return type str

pytoshop.util.**read_unicode_string**(*fd*)
Read a UTF-16-BE-encoded Unicode string (with length) from a file.

Parameters **fd** (*file-like object*) – Must be opened for reading, seekable and in binary mode.

Returns **value** – The unicode value of the string.

Return type str

pytoshop.util.**read_value**(*fd*, *fmt*, *endian='>'*)
Read a values from a file-like object.

Parameters

- **fd** (*file-like object*) – Must be opened for reading, in binary mode.
- **fmt** (*str*) – A `struct` module format character string.
- **endian** (*str*) – The endianness. Must be > or <. Default: >.

Returns

value – The value(s) read from the file.

If a single value, it is returned alone. If multiple values, a tuple is returned.

Return type any

pytoshop.util.**set_channel**(*color*, *channel*, *color_mode*, *channels*)

pytoshop.util.**trace_read**(*func*)
Prints debugging information from a read or write method.

For internal use only.

pytoshop.util.**trace_write**(*func*)
Prints debugging information from a read or write method.

For internal use only.

pytoshop.util.**unicode_string_length**(*value*)
Calculates the total length of writing a UTF-16-BE-encoded Unicode string (with length) to a file.

Parameters **value** (*str*) – A unicode string value.

Returns **length** – The length, in bytes.

Return type int

pytoshop.util.**unpack_bitflags**(*value*, *nbits*)
Unpack a bitfield into its constituent parts.

pytoshop.util.**write_pascal_string**(*fd*, *value*, *padding=1*)
Write a UTF-8-encoded Pascal string to a file.

Parameters

- **fd** (*file-like object*) – Must be opened for writing and in binary mode.
- **value** (*str*) – A unicode string value.
- **padding** (*int, optional*) – If provided, additional pad bytes will be written until the total amount written is a multiple of padding.

pytoshop.util.**write_unicode_string**(*fd*, *value*)
Write a UTF-16-BE-encoded Unicode string (with length) to a file.

Parameters

- **fd** (*file-like object*) – Must be opened for writing and in binary mode.
- **value** (*str*) – A unicode string value.

pytoshop.util.**write_value**(*fd*, *fmt*, **value*, ***kwargs*)
Write a single binary value to a file-like object.

Parameters

- **fd** (*file-like object*) – Must be opened for writing, in binary mode.
- **fmt** (*str*) – A `struct` module format character string.
- **value** (*any*) – The value to encode and write to the file.
- **endian** (*str*) – The endianness. Must be > or <. Default: >.

CHAPTER 5

Credits

5.1 Development Lead

- Michael Droettboom <mdboom@gmail.com>

Parts of this code were inspired by psd-tools, so a huge thanks to Mikhail Korobov and the rest of the psd-tools team.

5.2 Contributors

CHAPTER 6

History

6.1 1.2.1 (2018-11-30) (2017-12-13) (2017-09-26)

Bugfixes:

- Fix #4: Change masked layer to same size as image layer when using the *user.nested_layers* interface.

6.2 1.0.1 (2017-08-01) (2017-07-29)

- Declared API as stable.

6.3 0.6.0 (2017-07-24)

Minor improvements:

- Make it easier to choose the right channel, given the current color mode.

6.4 0.5.0 (2017-07-03)

Minor improvements:

- The color assigned to a layer (displayed in the layer list) is now available in the `nested_layers` API.

Bugfixes:

- Fix crash in `GuideResourceBlock`.

6.5 0.4.1 (2017-06-05)

Minor improvements:

- Parse multiple values in a struct together when possible. This has a modest speed improvement.

Bugfixes:

- Fix pip install [#1]
- The main canvas image will use the specified compression algorithm when using nested_layers_to_psd.
- Non-image layers correctly set "pixel_data_is_irrelevant" flag.

6.6 0.4.0 (2017-05-12)

Improvements:

- For speed purposes, pytoshop no longer uses traitlets.
- Performance improvements to the compression/decompression code.
- Added support for the `shmd` metadata tagged block, and the ability to access it from the `user.nested_layers` API.

Bugfixes:

- Updated the list of tagged blocks that use 8-bit lengths.
- Fixed a bug where the image data would be corrupted when writing images from an input file to an output file with a different file format version.
- Fixed a crash when the input file contains no layer group ids.
- Allow Numpy arrays of shape () in place of scalars for constant images.

6.7 0.3.0 (2017-01-09)

Improvements:

- `pytoshop` now runs on Python 2.7, in addition to 3.4 and 3.5.
- Many of the image resources types are now handled directly, rather than through a generic bytes-only interface.
- Major speedups in compression codecs.

Bugfixes:

- Saving a layer with a constant color (in `nested_layers_to_psd`) now works correctly.
- Unicode string decoding now properly handles trailing zeroes.
- The "name source" on layers (when created from `nested_layers_to_psd`) would point to the wrong source, but is now fixed.
- Fix a bug when writing a layer of width 1.

Python Module Index

p

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z