

---

# **jicbioimage.core Documentation**

***Release 0.15.0***

**Tjelvar Olsson and Matthew Hartley**

October 31, 2016



|          |                                                         |           |
|----------|---------------------------------------------------------|-----------|
| <b>1</b> | <b>The <code>jicbioimage.core</code> Python package</b> | <b>3</b>  |
| 1.1      | Features . . . . .                                      | 3         |
| 1.2      | Related packages . . . . .                              | 3         |
| <b>2</b> | <b>API documentation</b>                                | <b>5</b>  |
| 2.1      | <code>jicbioimage.core.image</code> . . . . .           | 5         |
| 2.2      | <code>jicbioimage.core.io</code> . . . . .              | 8         |
| 2.3      | <code>jicbioimage.core.transform</code> . . . . .       | 10        |
| 2.4      | <code>jicbioimage.core.util.array</code> . . . . .      | 10        |
| 2.5      | <code>jicbioimage.core.util.color</code> . . . . .      | 11        |
|          | <b>Python Module Index</b>                              | <b>13</b> |



jicimagelib documentation



---

## The `jicbioimage.core` Python package

---

The `jicbioimage.core` Python package provides the core functionality of the `jicbioimage` namespace package.

- Documentation: <http://jicbioimagecore.readthedocs.io>
- GitHub: <https://github.com/JIC-CSB/jicbioimage.core>
- PyPI: <https://pypi.python.org/pypi/jicbioimage.core>
- Free software: MIT License

### 1.1 Features

- Built in functionality for working with microscopy data
- Automatic generation of audit trails
- IPython integration
- Cross-platform: Linux, Mac and Windows are all supported
- Works with Python 2.7, 3.3 and 3.4

### 1.2 Related packages

#### 1.2.1 `jicbioimage`

- Documentation: <http://jicbioimage.readthedocs.io>
- GitHub: <https://github.com/JIC-CSB/jicbioimage>

#### 1.2.2 `jicbioimage.transform`

- Documentation: <http://jicbioimagetransform.readthedocs.io>
- GitHub: <https://github.com/JIC-CSB/jicbioimage.transform>

### 1.2.3 `jicbioimage.segment`

- Documentation: <http://jicbioimagesegment.readthedocs.io>
- GitHub: <https://github.com/JIC-CSB/jicbioimage.segment>

### 1.2.4 `jicbioimage.illustrate`

- Documentation: <http://jicbioimageillustrate.readthedocs.io>
- GitHub: <https://github.com/JIC-CSB/jicbioimage.illustrate>

---

API documentation

---

## 2.1 jicbioimage.core.image

Module for managing and accessing images.

**class** `jicbioimage.core.image.History` (*creation=None*)

Class for storing the provenance of an image.

**class** `Event` (*function, args, kwargs*)

An event in the history of an image.

`History.add_event` (*function, args, kwargs*)

Return event added to the history.

**class** `jicbioimage.core.image.Image` (*shape, dtype=<type 'numpy.uint8'>, buffer=None, offset=0, strides=None, order=None, name=None, log\_in\_history=True*)

Image class.

**classmethod** `from_file` (*fpath, name=None, log\_in\_history=True*)

Return `jicbioimage.core.image.Image` instance from a file.

**Parameters**

- **fpath** – path to the image file
- **name** – name of the image
- **log\_in\_history** – whether or not to log the creation event in the image's history

**Returns** `jicbioimage.core.image.Image`

**class** `jicbioimage.core.image.Image3D` (*shape, dtype=<type 'numpy.uint8'>, buffer=None, offset=0, strides=None, order=None, name=None, log\_in\_history=True*)

Image3D class; in other words a 3D stack.

**classmethod** `from_directory` (*directory*)

Return `jicbioimage.core.image.Image3D` from directory.

**Parameters** **directory** – name of input directory

**Returns** `jicbioimage.core.image.Image3D`

**to\_directory** (*directory*)

Write slices from 3D image to directory.

**write** (*name*)

Write slices from 3D image to disk.

**Parameters** **name** – name of output directory

**class** `jicbioimage.core.image.ImageCollection` (*fpath=None*)

Class for storing related images.

**image** (*index=0*)

Return image as a `jicbioimage.core.image.Image`.

**Parameters** **index** – list index

**Returns** `jicbioimage.core.image.Image`

**parse\_manifest** (*fpath*)

Parse manifest file to build up the collection of images.

**Parameters** **fpath** – path to the manifest file

**Raises** `RuntimeError`

**proxy\_image** (*index=0*)

Return a `jicbioimage.core.image.ProxyImage` instance.

**Parameters** **index** – list index

**Returns** `jicbioimage.core.image.ProxyImage`

**class** `jicbioimage.core.image.MicroscopyCollection` (*fpath=None*)

Collection of `jicbioimage.core.image.MicroscopyImage` instances.

**channels** (*s=0*)

Return list of channels in the collection.

**Parameters** **s** – series

**Returns** list of channel identifiers

**image** (*s=0, c=0, z=0, t=0*)

Return image as a `jicbioimage.core.image.Image`.

**Parameters**

- **s** – series
- **c** – channel
- **z** – zslicing
- **t** – timepoint

**Returns** `jicbioimage.core.image.Image`

**proxy\_image** (*s=0, c=0, z=0, t=0*)

Return a `jicbioimage.core.image.MicroscopyImage` instance.

**Parameters**

- **s** – series
- **c** – channel
- **z** – zslicing
- **t** – timepoint

**Returns** `jicbioimage.core.image.MicroscopyImage`

**series**

Return list of series in the collection.

**timepoints** (*s=0*)

Return list of time points in the collection.

**Parameters** *s* – series

**Returns** list of time point identifiers

**zslices** (*s=0*)

Return list of z-slices in the collection.

**Parameters** *s* – series

**Returns** list of zslice identifiers

**zstack** (*s=0, c=0, t=0*)

Return zstack as a *jicbioimage.core.image.Image3D*.

**Parameters**

- *s* – series
- *c* – channel
- *t* – timepoint

**Returns** zstack as a *jicbioimage.core.image.Image3D*

**zstack\_array** (*s=0, c=0, t=0*)

Return zstack as a *numpy.ndarray*.

**Parameters**

- *s* – series
- *c* – channel
- *t* – timepoint

**Returns** zstack as a *numpy.ndarray*

**zstack\_proxy\_iterator** (*s=0, c=0, t=0*)

Return zstack *jicbioimage.core.image.ProxyImage* iterator.

**Parameters**

- *s* – series
- *c* – channel
- *t* – timepoint

**Returns** zstack as a *jicbioimage.core.image.ProxyImage* iterator

**class** *jicbioimage.core.image.MicroscopyImage* (*fpath, metadata={}*)

Lightweight image class with microscopy meta data.

**in\_zstack** (*s, c, t*)

Return True if I am in the zstack.

**Parameters**

- *s* – series
- *c* – channel
- *t* – timepoint

**Returns** bool

**is\_me** (*s, c, z, t*)

Return True if arguments match my meta data.

**Parameters**

- **s** – series
- **c** – channel
- **z** – zslice
- **t** – timepoint

**Returns** bool

**class** jicbioimage.core.image.**ProxyImage** (*fpath, metadata={}*)

Lightweight image class.

**image**

Underlying *jicbioimage.core.image.Image* instance.

## 2.2 jicbioimage.core.io

Module for reading and writing images.

**class** jicbioimage.core.io.**AutoName**

Class for generating output file names automatically.

**directory** = None

Output directory to save images to.

**classmethod** **name** (*func*)

Return auto generated file name.

**namespace** = ''

Image file namespace.

**classmethod** **prefix** ()

Return auto generated file prefix.

**prefix\_format** = '{:d}\_'

Image file prefix format.

**class** jicbioimage.core.io.**AutoWrite**

Class for writing images automatically.

**on** = True

Whether or not auto writing of images is enabled.

**class** jicbioimage.core.io.**BFConvertWrapper** (*backend*)

Class for unpacking microscopy files using bfconvert.

**already\_converted** (*fpath*)

Return true if the file already has a manifest file in the backend.

**Parameters** **fpath** – potential path to the manifest file

**Returns** bool

**manifest** (*entry*)

Returns manifest as a list.

**Parameters** `entry` – `jicbioimage.core.io.FileBackend.Entry`

**Returns** `jicbioimage.core.io.Manifest`

**metadata\_from\_fname** (`fname`, `md5_hexdigest`)

Return meta data extracted from file name.

**Parameters** `fname` – metadata file name

**Returns** dictionary with meta data required by

**run\_command** (`input_file`, `output_dir=None`)

Return the command for running bfconvert as a list.

**Parameters**

- **input\_file** – path to microscopy image to be converted
- **ouput\_dir** – directory to write output tiff files to

**Returns** list

**split\_pattern** (`win32=False`)

Pattern used to split the input file.

**class** `jicbioimage.core.io.DataManager` (`backend=None`)

Manage `jicbioimage.core.image.ImageCollection` instances.

**load** (`fpath`)

Load a microscopy file.

**Parameters** `fpath` – path to microscopy file

**class** `jicbioimage.core.io.FileBackend` (`directory`)

Class for storing image files.

**class** `Entry` (`base_dir`, `fpath`)

Class representing a backend entry.

**directory**

Where the images are stored.

`FileBackend.directory`

Where the entries are stored.

`FileBackend.new_entry` (`fpath`)

Return a new entry; to be populated with images.

**Parameters** `fpath` – path to microscopy image

**Returns** `jicimaginglib.image.FileBackend.Entry` instance

**class** `jicbioimage.core.io.Manifest`

Class for generating backend entry manifest files.

**add** (`filename`, `**kwargs`)

Add an entry to the manifest.

**Parameters**

- **filename** – relative path to image
- **kwargs** – custom parameters, e.g. series, channel, zslice

**Returns** the added entry

**json**

Return json representation.

## 2.3 jicbioimage.core.transform

Module containing image transformation functions.

This module contains the function decorator `jicbioimage.core.transform.transformation()` that can be used to turn functions into image transformations.

Below is an example of how to create a transformation that inverts an image.

```
>>> import numpy as np
>>> @transformation
... def invert(image):
...     "Return the inverted image."
...     maximum = np.iinfo(image.dtype).max
...     maximum_array = np.ones(image.shape, dtype=image.dtype) * maximum
...     return maximum_array - image
... 
```

`jicbioimage.core.transform.transformation` (*func*)

Function decorator to turn another function into a transformation.

## 2.4 jicbioimage.core.util.array

Module containing utility functions for manipulating numpy arrays.

`jicbioimage.core.util.array.check_dtype` (*array*, *allowed*)

Raises `TypeError` if the array is not of an allowed dtype.

### Parameters

- **array** – array whose dtype is to be checked
- **allowed** – instance or list of allowed dtypes

**Raises** `TypeError`

`jicbioimage.core.util.array.color_array` (*array*, *color\_dict*)

Return RGB color array.

Assigning a unique RGB color value to each unique element of the input array and return an array of shape (`array.shape`, 3).

### Parameters

- **array** – input `numpy.array`
- **color\_dict** – dictionary with keys/values corresponding to identifiers and RGB tuples respectively

`jicbioimage.core.util.array.dtype_contract` (*input\_dtype=None*, *output\_dtype=None*)

Function decorator for specifying input and/or output array dtypes.

### Parameters

- **input\_dtype** – dtype of input array
- **output\_dtype** – dtype of output array

**Returns** function decorator

`jicbioimage.core.util.array.map_stack(array3D, z_function)`

Return 3D array where each z-slice has had the function applied to it.

**Parameters**

- **array3D** – 3D numpy.array
- **z\_function** – function to be mapped to each z-slice

`jicbioimage.core.util.array.normalise(array)`

Return array normalised such that all values are between 0 and 1.

If all the values in the array are the same the function will return: - `np.zeros(array.shape, dtype=np.float)` if the value is 0 or less - `np.ones(array.shape, dtype=np.float)` if the value is greater than 0

**Parameters** **array** – numpy.array

**Returns** `numpy.array.astype(numpy.float)`

`jicbioimage.core.util.array.pretty_color_array(array, keep_zero_black=True)`

Return a RGB pretty color array.

Assigning a pretty RGB color value to each unique element of the input array and return an array of shape `(array.shape, 3)`.

**Parameters**

- **array** – input numpy.array
- **keep\_zero\_black** – whether or not the background should be black

**Returns** `numpy.array`

`jicbioimage.core.util.array.reduce_stack(array3D, z_function)`

Return 2D array projection of the input 3D array.

The input function is applied to each line of an input x, y value.

**Parameters**

- **array3D** – 3D numpy.array
- **z\_function** – function to use for the projection (e.g. `max()`)

`jicbioimage.core.util.array.unique_color_array(array)`

Return a RGB unique color array.

Assigning a unique RGB color value to each unique element of the input array and return an array of shape `(array.shape, 3)`.

**Parameters** **array** – input numpy.array

**Returns** `numpy.array`

## 2.5 jicbioimage.core.util.color

Module for generating RGB tuples for use as colors in images.

`jicbioimage.core.util.color.identifier_from_unique_color(unique_color)`

Return identifier from unique RGB tuple.

**Parameters** **unique\_color** – RGB tuple

**Returns** positive integer in range from 0 to 16777215 inclusive

`jicbioimage.core.util.color.pretty_color_from_identifier(identifier)`  
Return deterministic aesthetically pleasing RGB tuple.

**Returns** RGB tuple

`jicbioimage.core.util.color.pretty_color_palette(identifiers, keep_zero_black=True)`  
Return dictionary with pretty colors.

**Parameters**

- **identifiers** – set of unique identifiers
- **keep\_zero\_black** – whether or not the background should be black

**Returns** dictionary

`jicbioimage.core.util.color.random_pretty_color()`  
Return random aesthetically pleasing RGB tuple.

**Returns** RGB tuple

`jicbioimage.core.util.color.unique_color_from_identifier(identifier)`  
Return unique color as RGB tuple.

Useful for creating PNG images where each color is used as an identifier.

Raises `TypeError` if the identifier is not an integer.

Raises `ValueError` if the identifier is not in the range 0 to 16777215 inclusive.

**Parameters** **identifier** – positive integer in range from 0 to 16777215 inclusive

**Raises** `TypeError`, `ValueError`

**Returns** RGB tuple

`jicbioimage.core.util.color.unique_color_palette(identifiers)`  
Return dictionary with unique colors.

**Parameters** **identifiers** – set of unique identifiers

**Returns** dictionary

## j

`jicbioimage.core.image`, [5](#)  
`jicbioimage.core.io`, [8](#)  
`jicbioimage.core.transform`, [10](#)  
`jicbioimage.core.util.array`, [10](#)  
`jicbioimage.core.util.color`, [11](#)



**A**

add() (jicbioimage.core.io.Manifest method), 9  
 add\_event() (jicbioimage.core.image.History method), 5  
 already\_converted() (jicbioimage.core.io.BFConvertWrapper method), 8  
 AutoName (class in jicbioimage.core.io), 8  
 AutoWrite (class in jicbioimage.core.io), 8

**B**

BFConvertWrapper (class in jicbioimage.core.io), 8

**C**

channels() (jicbioimage.core.image.MicroscopyCollection method), 6  
 check\_dtype() (in module jicbioimage.core.util.array), 10  
 color\_array() (in module jicbioimage.core.util.array), 10

**D**

DataManager (class in jicbioimage.core.io), 9  
 directory (jicbioimage.core.io.AutoName attribute), 8  
 directory (jicbioimage.core.io.FileBackend attribute), 9  
 directory (jicbioimage.core.io.FileBackend.Entry attribute), 9  
 dtype\_contract() (in module jicbioimage.core.util.array), 10

**F**

FileBackend (class in jicbioimage.core.io), 9  
 FileBackend.Entry (class in jicbioimage.core.io), 9  
 from\_directory() (jicbioimage.core.image.Image3D class method), 5  
 from\_file() (jicbioimage.core.image.Image class method), 5

**H**

History (class in jicbioimage.core.image), 5  
 History.Event (class in jicbioimage.core.image), 5

**I**

identifier\_from\_unique\_color() (in module jicbioimage.core.util.color), 11  
 Image (class in jicbioimage.core.image), 5  
 image (jicbioimage.core.image.ProxyImage attribute), 8  
 image() (jicbioimage.core.image.ImageCollection method), 6  
 image() (jicbioimage.core.image.MicroscopyCollection method), 6  
 Image3D (class in jicbioimage.core.image), 5  
 ImageCollection (class in jicbioimage.core.image), 6  
 in\_zstack() (jicbioimage.core.image.MicroscopyImage method), 7  
 is\_me() (jicbioimage.core.image.MicroscopyImage method), 8

**J**

jicbioimage.core.image (module), 5  
 jicbioimage.core.io (module), 8  
 jicbioimage.core.transform (module), 10  
 jicbioimage.core.util.array (module), 10  
 jicbioimage.core.util.color (module), 11  
 json (jicbioimage.core.io.Manifest attribute), 9

**L**

load() (jicbioimage.core.io.DataManager method), 9

**M**

Manifest (class in jicbioimage.core.io), 9  
 manifest() (jicbioimage.core.io.BFConvertWrapper method), 8  
 map\_stack() (in module jicbioimage.core.util.array), 11  
 metadata\_from\_fname() (jicbioimage.core.io.BFConvertWrapper method), 9  
 MicroscopyCollection (class in jicbioimage.core.image), 6  
 MicroscopyImage (class in jicbioimage.core.image), 7

**N**

name() (jicbioimage.core.io.AutoName class method), 8

namespace (jicbioimage.core.io.AutoName attribute), 8  
new\_entry() (jicbioimage.core.io.FileBackend method), 9  
normalise() (in module jicbioimage.core.util.array), 11

## O

on (jicbioimage.core.io.AutoWrite attribute), 8

## P

parse\_manifest() (jicbioimage.core.image.ImageCollection method), 6  
prefix() (jicbioimage.core.io.AutoName class method), 8  
prefix\_format (jicbioimage.core.io.AutoName attribute), 8  
pretty\_color\_array() (in module jicbioimage.core.util.array), 11  
pretty\_color\_from\_identifier() (in module jicbioimage.core.util.color), 12  
pretty\_color\_palette() (in module jicbioimage.core.util.color), 12  
proxy\_image() (jicbioimage.core.image.ImageCollection method), 6  
proxy\_image() (jicbioimage.core.image.MicroscopyCollection method), 6  
ProxyImage (class in jicbioimage.core.image), 8

## R

random\_pretty\_color() (in module jicbioimage.core.util.color), 12  
reduce\_stack() (in module jicbioimage.core.util.array), 11  
run\_command() (jicbioimage.core.io.BFCConvertWrapper method), 9

## S

series (jicbioimage.core.image.MicroscopyCollection attribute), 6  
split\_pattern() (jicbioimage.core.io.BFCConvertWrapper method), 9

## T

timepoints() (jicbioimage.core.image.MicroscopyCollection method), 7  
to\_directory() (jicbioimage.core.image.Image3D method), 5  
transformation() (in module jicbioimage.core.transform), 10

## U

unique\_color\_array() (in module jicbioimage.core.util.array), 11  
unique\_color\_from\_identifier() (in module jicbioimage.core.util.color), 12

unique\_color\_palette() (in module jicbioimage.core.util.color), 12

## W

write() (jicbioimage.core.image.Image3D method), 5

## Z

zslices() (jicbioimage.core.image.MicroscopyCollection method), 7  
zstack() (jicbioimage.core.image.MicroscopyCollection method), 7  
zstack\_array() (jicbioimage.core.image.MicroscopyCollection method), 7  
zstack\_proxy\_iterator() (jicbioimage.core.image.MicroscopyCollection method), 7