

---

# **dicomslide Documentation**

***Release 0.7.1***

**Markus D. Herrmann**

**Dec 29, 2022**



## CONTENTS:

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                           | <b>1</b>  |
| 1.1      | Design . . . . .                              | 1         |
| 1.2      | Application programming interface . . . . .   | 1         |
| <b>2</b> | <b>Installation guide</b>                     | <b>3</b>  |
| 2.1      | Requirements . . . . .                        | 3         |
| 2.2      | Installation . . . . .                        | 3         |
| <b>3</b> | <b>User guide</b>                             | <b>5</b>  |
| 3.1      | Constructing a DICOM Client . . . . .         | 5         |
| 3.2      | Reading images using dicomslide API . . . . . | 5         |
| 3.3      | Reading images using openslide API . . . . .  | 6         |
| <b>4</b> | <b>Developer guide</b>                        | <b>7</b>  |
| 4.1      | Pull requests . . . . .                       | 7         |
| 4.2      | Coding style . . . . .                        | 7         |
| 4.3      | Running tests . . . . .                       | 7         |
| 4.4      | Building documentation . . . . .              | 8         |
| <b>5</b> | <b>License</b>                                | <b>9</b>  |
| <b>6</b> | <b>API Documentation</b>                      | <b>11</b> |
| 6.1      | dicomslide package . . . . .                  | 11        |
| <b>7</b> | <b>Indices and tables</b>                     | <b>39</b> |
|          | <b>Python Module Index</b>                    | <b>41</b> |
|          | <b>Index</b>                                  | <b>43</b> |



## INTRODUCTION

The `dicomslide` build distribution provides an application programming interface (API) for querying and retrieving whole slide images in DICOM format from local files or over network via a unified application programming interface.

The `dicomslide` Python package contains several classes and functions.

### 1.1 Design

The `dicomslide` Python package contains several data structures that abstract whole slide images. The core data structure of the library is the `dicomslide.Slide` class, which represents a collection of `DICOM VL Whole Slide Microscopy Image` instances that share the same Container Identifier and Frame of Reference UID, i.e., that were acquired for the same physical class slide and are spatially aligned. The interface exposed by the `dicomslide.Slide` class abstracts the organization of tiled images that belong to the slide (`dicomslide.TiledImage`) and the associated total pixel matrices (`dicomslide.TotalPixelMatrix`), which form a multi-resolution image pyramid (`dicomslide.Pyramid`).

### 1.2 Application programming interface

The library leverages the Python `dicomweb-client` library to efficiently search for and retrieve whole slide image data from heterogeneous sources using the interface defined by the `dicomweb_client.DICOMClient` protocol. Importantly, the library does not load the entire images into memory, but dynamically retrieves only the image frames (tiles) that are needed for a requested image region.

The `dicomweb_client.DICOMwebClient` and `dicomweb_client.DICOMfileClient` classes both implement that protocol and thereby enable efficient reading of whole slide images in DICOM format from remote archives using the DICOMweb RESTful API (see [DICOM Part 18](#)) and from local DICOM files (see [DICOM Part 10](#)), respectively.



## INSTALLATION GUIDE

### 2.1 Requirements

- [Python](#) (version 3.6 or higher)
- Python package manager [pip](#)

### 2.2 Installation

Pre-build package available at PyPi:

```
pip install dicomslide
```

Source code available at Github:

```
git clone https://github.com/herrmannlab/highdicom ~/highdicom  
pip install ~/highdicom
```





Reading whole slide images in DICOM format using the *dicomslide* package.

## 3.1 Constructing a DICOM Client

Use `dicomweb_client.DICOMfileClient` to read whole slide images from DICOM files stored on a file system:

```
import dicomweb_client

client = dicomweb_client.DICOMfileClient(url='file:///tmp/images')
```

Use `dicomweb_client.DICOMwebClient` to read whole slide images over network using DICOMweb services:

```
import dicomweb_client

client = dicomweb_client.DICOMwebClient(url='http://myserver.com/dicomweb')
```

## 3.2 Reading images using dicomslide API

```
import dicomslide
import numpy as np
from matplotlib import pyplot as plt

found_slides = dicomslide.find_slides(client, container_id='S22-ABC-123')
assert len(found_slides) == 1
slide = found_slides[0]

print(slide.num_channels)
print(slide.num_focal_planes)
print(slide.num_levels)
print(slide.total_pixel_matrix_dimensions)
print(slide.downsampling_factors)
print(slide.label_images)
print(slide.get_volume_images(channel_index=0, focal_plane_index=0))

region: np.ndarray = slide.get_image_region(
    pixel_indices=(0, 0),
    level=-1,
```

(continues on next page)

(continued from previous page)

```
    size=(512, 512),
    channel_index=0,
    focal_plane_index=0
)
plt.imshow(region)
plt.show()
```

### 3.3 Reading images using openslide API

The library also exposes an `OpenSlide` interface (`dicomslide.OpenSlide`), which is intended as an API wrapper around a `dicomslide.Slide` instance and to be used as a drop-in replacement for an `openslide.OpenSlide` instance:

```
from PIL import Image

openslide = dicomslide.OpenSlide(slide)

print(openslide.level_count)
print(openslide.dimensions)
print(openslide.level_dimensions)
print(openslide.level_downsamples)
print(openslide.properties)
print(openslide.associated_images)

thumbnail: Image.Image = openslide.get_thumbnail(size=(50, 100))
thumbnail.show()
```

Note that the `OpenSlide` API only supports 2D color images. For images with multiple channels or Z-planes, only the standard `dicomslide` API can be used.

## DEVELOPER GUIDE

Source code is available at Github and can be cloned via git:

```
git clone https://github.com/herrmannlab/dicomslide ~/dicomslide
```

The *dicomslide* package can be installed in *develop* mode for local development:

```
pip install -e ~/dicomslide
```

### 4.1 Pull requests

Don't commit code changes to the `master` branch. New features should be implemented in a separate branch called `feature/*` and bug fixes should be applied in separate branch called `bugfix/*`.

Before creating a pull request on Github, read the coding style guideline, run the tests and check PEP8 compliance.

### 4.2 Coding style

Code must comply with [PEP 8](#). The `flake8` package is used to enforce compliance.

The project uses `numpydoc` for documenting code according to [PEP 257](#) docstring conventions. Further information and examples for the NumPy style can be found at the [NumPy Github repository](#) and the website of the [Napoleon sphinx extension](#).

All API classes, functions and modules must be documented (including “private” functions and methods). Each docstring must describe input parameters and return values. Types must be specified using type hints as specified by [PEP 484](#) (see `typing` module) in both the function definition as well as the docstring.

### 4.3 Running tests

The project uses `pytest` to write and runs unit tests. Tests should be placed in a separate `tests` folder within the package root folder. Files containing actual test code should follow the pattern `test_*.py`.

Install requirements:

```
pip install -r ~/dicomslide/requirements_test.txt
```

Run tests (including checks for PEP8 compliance):

```
cd ~/dicomslide
pytest --flake8
```

## 4.4 Building documentation

Install requirements:

```
pip install -r ~/dicomslide/requirements_docs.txt
```

Build documentation in *HTML* format:

```
cd ~/dicomslide
make html
```

The built `index.html` file will be located in `docs/build`.

## LICENSE

*dicomslide* is free and open source software licensed under the permissive [MIT license](#).



## API DOCUMENTATION

### 6.1 dicomslide package

**class** dicomslide.ChannelTypes(*value*)

Bases: Enum

Enumerated values for channel types.

OPTICAL\_PATH = 'OPTICAL\_PATH'

PARAMETER = 'PARAMETER'

SEGMENT = 'SEGMENT'

**class** dicomslide.ImageFlavors(*value*)

Bases: Enum

Enumerated values for image flavors.

LABEL = 'LABEL'

OVERVIEW = 'OVERVIEW'

THUMBNAIL = 'THUMBNAIL'

VOLUME = 'VOLUME'

**class** dicomslide.OpenSlide(*slide*)

Bases: object

Wrapper class that exposes data of a slide via the OpenSlide interface.

---

**Note:** There are two major differences between the OpenSlide interface exposed by this class and the interface exposed by [\*dicomslide.Slide\*](#):

1. The OpenSlide API returns images as :class:`PIL.Image.Image` objects, while :class:`dicomslide.Slide` returns pixel arrays as :class:`numpy.ndarray` objects.
  2. The OpenSlide API specifies image dimensions and indices in column-major order (following the Pillow convention), while :class:`dicomslide.Slide` specifies array dimensions and indices in row-major order (following the NumPy convention).
-

**Parameters**

**slide** (`dicomslide.slide.Slide`) – DICOM slide

**property associated\_images:** `Dict[str, Image]`

Mapping of image flavor (LABEL or OVERVIEW) to image

**Type**

`Dict[str, PIL.Image.Image]`

**Return type**

`typing.Dict[str, PIL.Image.Image]`

**close()**

**Return type**

`None`

**property dimensions:** `Tuple[int, int]`

Width and height of images at base level 0

**Type**

`Tuple[int, int]`

**Return type**

`typing.Tuple[int, int]`

**get\_best\_level\_for\_downsample**(*downsample*)

Compute best level for displaying the given downsample.

**Parameters**

**downsample** (*float*) – Desired downsample factor

**Returns**

Zero-based level index

**Return type**

`int`

**get\_thumbnail**(*size*)

Create a thumbnail of the slide.

**Parameters**

**size** (*`Tuple[int, int]`*) – Number of pixels columns and rows that the thumbnail should have

**Returns**

RGB image

**Return type**

`PIL.Image.Image`

**property level\_count:** `int`

Number of pyramid resolution levels

**Type**

`int`

**Return type**

`int`



**property level\_dimensions:** `Tuple[Tuple[int, int], ...]`

Width and height of images at each level

**Type**

`Tuple[Tuple[int, int]]`

**Return type**

`typing.Tuple[typing.Tuple[int, int], ...]`

**property level\_downsamples:** `Tuple[float, ...]`

Downsampling factor of images at each level with respect to the base level 0

**Type**

`Tuple[float]`

**Return type**

`typing.Tuple[float, ...]`

**property properties:** `Dict[str, str]`

Metadata about the slide.

**Returns**

OpenSlide properties

**Return type**

`Dict[str, str]`

**read\_region**(*location, level, size*)

Read region of a VOLUME (or THUMBNAIL) image at a given level.

**Parameters**

- **location** (`Tuple[int, int]`) – Zero-based (column, row) offset of the region from the topleft hand pixel of of the total pixel matrix of the image at the base level 0
- **level** (`int`) – Zero-based level index
- **size** (`Tuple[int, int]`) – Number of pixels columns and rows that should be read from the total pixel matrix at the specified level

**Returns**

RGBA image

**Return type**

`PIL.Image.Image`

**class** `dicomslide.Pyramid`(*metadata, tolerance, ref\_metadata=None*)

Bases: `object`

Image pyramid.

**Parameters**

- **metadata** (`Sequence[pydicom.Dataset]`) – Metadata of DICOM image instances
- **tolerance** (`float`) – Maximally tolerated distances between the centers of images at different pyramid levels in the slide coordinate system in millimeter unit
- **ref\_metadata** (`Union[Sequence[pydicom.Dataset], None]`, *optional*) – Metadata of referenced DICOM source image instances that may serve as a template

```
class dicomslide.PyramidLevel(total_pixel_matrix_dimensions: Tuple[int, int], pixel_spacing: Tuple[float, float], downsampling_factors: Tuple[float, float], has_pixels: bool)
```

Bases: tuple

Image pyramid level.

Create new instance of PyramidLevel(*total\_pixel\_matrix\_dimensions*, *pixel\_spacing*, *downsampling\_factors*, *has\_pixels*)

**property** *downsampling\_factors*

Alias for field number 2

**property** *has\_pixels*

Alias for field number 3

**property** *pixel\_spacing*

Alias for field number 1

**property** *total\_pixel\_matrix\_dimensions*

Alias for field number 0

```
class dicomslide.Slide(client, image_metadata, max_frame_cache_size=6, pyramid_tolerance=0.1)
```

Bases: object

A digital slide.

A collection of DICOM image instances that share the same Frame of Reference UID and Container Identifier, i.e., that have been acquired as part of one image acquisition for the same physical glass slide (container) and can be visualized and analyzed in the same frame of reference (coordinate system).

A slide consists of one or more image pyramids - one for each unique pair of channel and focal plane. The total pixel matrices of the different pyramid levels are stored in separate DICOM image instances. Individual channels or focal planes may be each stored in separate DICOM image instances or combined in a single DICOM image instance per pyramid level. Pyramids are expected to have the same number of levels and the same downsampling factors across channels and focal planes and the total pixel matrices at each level are expected to have the same dimensions (i.e., the same number of total pixel matrix columns and rows). However, the tiling of the total pixel matrices (i.e., the number of tile columns and rows) may differ across pyramid levels as well as across channels and focal planes at the same pyramid level.

#### Parameters

- **client** (*dicomweb\_client.api.DICOMClient*) – DICOMweb client
- **image\_metadata** (*Sequence[pydicom.Dataset]*) – Metadata of DICOM VL Whole Slide Microscopy Image instances or of derived DICOM Segmentation or Parametric Map instances that belong to the slide
- **max\_frame\_cache\_size** (*int, optional*) – Maximum number of frames that should be cached per image instance to avoid repeated retrieval requests
- **pyramid\_tolerance** (*float, optional*) – Maximally tolerated distances between the centers of images at different pyramid levels in the slide coordinate system in millimeter unit

**property** *downsampling\_factors: Tuple[float, ...]*

Downsampling factors of images at each pyramid level relative to the base level

#### Type

Tuple[float]

#### Return type

typing.Tuple[float, ...]

**find\_optical\_paths**(*identifier=None, description=None, illumination\_wavelength=None, specimen\_stain=None*)

Find optical paths.

**Parameters**

- **identifier** (*Union[str, None], optional*) – Optical path identifier
- **description** (*Union[str, None], optional,*) – Optical path description
- **illumination\_wavelength** (*Union[float, None], optional,*) – Optical path illumination wavelength
- **specimen\_stain** (*Union[hd.sr.CodedConcept, Code, None], optional*) – Substance used for specimen staining

**Returns**

Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMBNAIL images.

**Return type**

Tuple[int, ...]

**find\_segments**(*number=None, label=None, property\_category=None, property\_type=None*)

Find segments.

**Parameters**

- **number** (*Union[int, None], optional*) – Segment number
- **label** (*Union[str, None], optional,*) – Segment label
- **property\_category** (*Union[hd.sr.CodedConcept, Code, None], optional*) – Category of segmented property
- **property\_type** (*Union[hd.sr.CodedConcept, Code, None], optional*) – Type of segmented property

**Returns**

Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMBNAIL images.

**Return type**

Tuple[int, ...]

**property\_frame\_of\_reference\_uid:** str

Unique identifier of the frame of reference

**Type**

str

**Return type**

str

**get\_channel\_identifier**(*channel\_index*)

Get identifier of a channel.

**Parameters**

**channel\_index** (*int*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMBNAIL images.

**Returns**

Channel identifier

**Return type**

str

**Raises****ValueError** – When no channel is found for *channel\_index***get\_channel\_index**(*channel\_identifier*, *channel\_type*)

Get index of a channel.

**Parameters**

- **channel\_identifier** (*str*) – Channel identifier
- **channel\_type** (*Union[str, dicomslide.ChannelTypes]*) – Channel type

**Returns**

Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute, which is dependend on the type of channel.

**Return type**

int

**Raises****ValueError** – When no channel is found for *channel\_identifier* and *channel\_type***get\_channel\_type**(*channel\_index*)

Get type of a channel.

**Parameters****channel\_index** (*int*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMBNAIL images.**Returns**

Channel type

**Return type***dicomslide.ChannelTypes***Raises****ValueError** – When no channel is found for *channel\_index***get\_focal\_plane\_index**(*focal\_plane\_offset*)

Get index of a focal plane.

**Parameters****focal\_plane\_offset** (*float*) – Offset of the focal plane from the from the slide surface along the Z axis of the slide coordinate system in micrometers**Returns**

Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Return type**

int

**Raises****ValueError** – When no focal plane is found for *focal\_plane\_offset***get\_focal\_plane\_offset**(*focal\_plane\_index*)

Get z offset of focal plane in slide coordinate system.

**Parameters**

**focal\_plane\_index** (*int*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Returns**

Offset of the focal plane from the from the slide surface along the Z axis of the slide coordinate system in micrometers

**Return type**

float

**Raises**

**ValueError** – When no focal plane is found for *focal\_plane\_index*

**get\_image\_region**(*offset, level, size, channel\_index=0, focal\_plane\_index=0*)

Get image region.

**Parameters**

- **offset** (*Tuple[int, int]*) – Zero-based (row, column) indices in the range [0, Rows) and [0, Columns), respectively, that specify the offset of the image region in the total pixel matrix of the image at the highest resolution level. The (0, 0) coordinate is located at the center of the topleft hand pixel in the total pixel matrix.
- **level** (*int*) – Zero-based index into pyramid levels
- **size** (*Tuple[int, int]*) – Rows and columns of the requested image region
- **channel\_index** (*int, optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMBNAIL images.
- **focal\_plane\_index** (*int, optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Returns**

Three-dimensional pixel array of shape (Rows, Columns, Samples per Pixel) for the requested image region

**Return type**

numpy.ndarray

**get\_pixel\_indices**(*offset, level, channel\_index=0, focal\_plane\_index=0*)

Get indices into total pixel matrix for a given slide position.

**Parameters**

- **offset** (*Tuple[float, float]*) – Zero-based (x, y) offset in the slide coordinate system in millimeter
- **level** (*int*) – Zero-based index into pyramid levels
- **channel\_index** (*int, optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMBNAIL images.
- **focal\_plane\_index** (*int, optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Returns**

Zero-based (row, column) position in the total pixel matrix of the image

**Return type**

Tuple[int, int]

---

**Note:** Pixel position may be negativ or extend beyond the size of the total pixel matrix if slide position at *offset* does fall into a region on the slide that was not imaged.

---

**get\_slide\_offset**(*pixel\_indices*, *level*, *channel\_index*=0, *focal\_plane\_index*=0)

Get slide coordinates for a given total pixel matrix position.

**Parameters**

- **pixel\_indices** (Tuple[int, int]) – Zero-based (row, column) offset in the total pixel matrix
- **level** (int) – Zero-based index into pyramid levels
- **channel\_index** (int, optional) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMB-NAIL images.
- **focal\_plane\_index** (int, optional) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Returns**

Zero-based (x, y) position on the slide in the slide coordinate system in millimeter

**Return type**

Tuple[float, float]

**get\_slide\_region**(*offset*, *level*, *size*, *channel\_index*=0, *focal\_plane\_index*=0)

Get slide region.

**Parameters**

- **offset** (Tuple[float, float]) – Zero-based (x, y) offset in the slide coordinate system in millimeter resolution. The (0.0, 0.0) coordinate is located at the origin of the slide (usually the slide corner).
- **level** (int) – Zero-based index into pyramid levels
- **size** (Tuple[float, float]) – Width and height of the requested slide region in millimeter unit along the X and Y axis of the slide coordinate system, respectively.
- **channel\_index** (int, optional) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMB-NAIL images.
- **focal\_plane\_index** (int, optional) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Returns**

Three-dimensional pixel array of shape (Rows, Columns, Samples per Pixel) for the requested slide region

**Return type**

numpy.ndarray

**Note:** The slide coordinate system is defined for the upright standing slide such that the X axis corresponds to the short side of the slide and the Y axis corresponds to the long side of the slide. The rows of the returned pixel array are thus parallel to the X axis of the slide coordinate system and the columns parallel to the Y axis of the slide coordinate system.

---

**get\_slide\_region\_for\_annotation**(*annotation, level, channel\_index=0, padding=0.0*)

Get slide region defined by a graphic annotation.

**Parameters**

- **annotation** (*highdicom.sr.Scoord3DContentItem*) – Graphic annotation that defines the region of interest (ROI) in the slide coordinate system
- **level** (*int*) – Zero-based index into pyramid levels
- **channel\_index** (*int, optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMB-NAIL images.
- **padding** (*Union[int, Tuple[int, int], Tuple[int, int, int, int]], optional*) – Padding on each border of the region defined by *annotation*. If a single integer is provided, the value is used to pad all four borders with the same number of pixels. If a sequence of length 2 is provided, the two values are used to pad the left/right (along the X axis) and top/bottom (along the Y axis) border, respectively. If a sequence of length 4 is provided, the four values are used to pad the left (- X axis), top (+ Y axis), right (+ X axis), and bottom (- Y axis) borders respectively.

**Returns**

Three-dimensional pixel array of shape (Rows, Columns, Samples per Pixel) for the requested slide region

**Return type**

numpy.ndarray

---

**Note:** The slide coordinate system is defined for the upright standing slide such that the X axis corresponds to the short side of the slide and the Y axis corresponds to the long side of the slide. The rows of the returned pixel array are thus parallel to the X axis of the slide coordinate system and the columns parallel to the Y axis of the slide coordinate system.

---

**get\_volume\_images**(*channel\_index=0, focal\_plane\_index=0*)

Get VOLUME or THUMBNAIL images for an channel and focal plane.

**Parameters**

- **channel\_index** (*int, optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMB-NAIL images.
- **focal\_plane\_index** (*int, optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMBNAIL images.

**Returns**

Images sorted by size in descending order

**Return type**

Tuple[*dicomslide.TiledImage, ...*]

**property label\_images:** `Tuple[TiledImage, ...]`

LABEL images of the slide

**Type**

`Tuple[dicomslide.TiledImage, ...]`

**Return type**

`typing.Tuple[dicomslide.image.TiledImage, ...]`

**map\_pixel\_indices\_to\_slide\_coordinates**(*pixel\_indices*, *level*, *channel\_index*=0,  
*focal\_plane\_index*=0)

Map pixel indices to slide coordinates.

**Parameters**

- **pixel\_indices** (*numpy.ndarray*) – Zero-based (row, column) indices into the total pixel matrix of the image
- **level** (*int*) – Zero-based index into pyramid levels
- **channel\_index** (*int*, *optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMB-NAIL images.
- **focal\_plane\_index** (*int*, *optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMB-NAIL images.

**Returns**

Zero-based (x, y, z) coordinates in the slide coordinate system in millimeter

**Return type**

`numpy.ndarray`

**map\_slide\_coordinates\_to\_pixel\_indices**(*slide\_coordinates*, *level*, *channel\_index*=0,  
*focal\_plane\_index*=0)

Map slide coordinates to pixel indices.

**Parameters**

- **slide\_coordinates** (*numpy.ndarray*) – Zero-based (x, y, z) coordinates in the slide coordinate system in millimeter
- **level** (*int*) – Zero-based index into pyramid levels
- **channel\_index** (*int*, *optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute of VOLUME or THUMB-NAIL images.
- **focal\_plane\_index** (*int*, *optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute of VOLUME or THUMB-NAIL images.

**Returns**

Zero-based (row, column) indices into the total pixel matrix of the image

**Return type**

`numpy.ndarray`

**property num\_channels:** `int`

Number of channels



**Type**  
int

**Return type**  
int

**property num\_focal\_planes:** int

Number of focal planes

**Type**  
int

**Return type**  
int

**property num\_levels:** int

Number of pyramid levels

---

**Note:** Levels are sorted by size in descending order from the base level (highest image resolution, smallest pixel spacing) to the top level (lowest image resolution, largest pixel spacing).

---

**Type**  
int

**Return type**  
int

**property overview\_images:** Tuple[TiledImage, ...]

OVERVIEW images of the slide

**Type**  
Tuple[dicomslide.TiledImage, ...]

**Return type**  
typing.Tuple[dicomslide.image.TiledImage, ...]

**property physical\_offset:** Tuple[float, float]

Minimum offset of the total pixel matrices from the origin of the frame of reference along the X and Y axes of the slide coordinate system in millimeter

**Type**  
Tuple[float, float]

**Return type**  
typing.Tuple[float, float]

**property physical\_size:** Tuple[float, float]

Maximum size of the total pixel matrices along the X and Y axes of the slide coordinate system in millimeter

**Type**  
Tuple[float, float]

**Return type**  
typing.Tuple[float, float]

**property pixel\_spacings:** Tuple[Tuple[float, float], ...]

Distance between neighboring pixels along the row (left to right) and column (top to bottom) directions

**Type**

Tuple[Tuple[float, float], ...]

**Return type**

typing.Tuple[typing.Tuple[float, float], ...]

**property size: Tuple[int, int]**

Maximum size of the total pixel matrices along the rows and columns axes of the total pixel matrix

**Type**

Tuple[int, int]

**Return type**

typing.Tuple[int, int]

**property total\_pixel\_matrix\_dimensions: Tuple[Tuple[int, int], ...]**

Number of columns and rows in the total pixel matrix for images at each pyramid level

**Type**

Tuple[Tuple[int, int], ...]

**Return type**

typing.Tuple[typing.Tuple[int, int], ...]

**class dicomslide.TiledImage(*client, image\_metadata, max\_frame\_cache\_size=6*)**

Bases: object

A tiled DICOM image.

An instance of the class represents a tiled DICOM image instance and provides methods for convenient and efficient access of image metadata and pixel data from a DICOMweb server (or another source for which the `dicomweb_client.DICOMClient` protocol has been implemented).

A tiled image is hereby defined as a DICOM image instance that contains the Total Pixel Matrix Rows and Total Pixel Matrix Columns attributes.

The class is designed to be independent of a particular DICOM Information Object Definition (IOD) or SOP Class and support various different types of DICOM images, including VL Whole Slide Microscopy Image, Segmentation, and Parametric Map.

Each image is associated with one or more `dicomslide.TotalPixelMatrix` instances, one for each unique combination of channel and focal plane. The definition of a channel is specific to a particular IOD. For example, in case of VL Whole Slide Microscopy Image, a channel corresponds to an optical path, whereas in case of a Segmentation, a channel corresponds to a segment.

## Examples

```
>>> image = TiledImage(...)
>>> print(image.metadata) # pydicom.Dataset
>>> print(image.metadata.BitsAllocated)
>>> print(image.metadata.TotalPixelMatrixRows)
>>> pixel_matrix = image.get_total_pixel_matrix(channel_index=0)
>>> print(pixel_matrix.dtype)
>>> print(pixel_matrix.shape)
>>> print(pixel_matrix[:1000, 350:750, :]) # numpy.ndarray
```

Construct object.

**Parameters**

- **client** (*dicomweb\_client.api.DICOMClient*) – DICOMweb client
- **image\_metadata** (*pydicom.dataset.Dataset*) – Metadata of a tiled DICOM image
- **max\_frame\_cache\_size** (*int, optional*) – Maximum number of frames that should be cached to avoid repeated retrieval requests

---

**Note:** If *image\_metadata* is the metadata of a color image, it should contain the ICC Profile element to enable color management. The value of this element may be considered bulkdata and therefore may have to be retrieved separately over DICOMweb.

---

**property channel\_type:** *ChannelTypes*

type of channels

**Type**

*dicomslide.ChannelTypes*

**Return type**

*dicomslide.enum.ChannelTypes*

**property frame\_of\_reference\_uid:** *str*

Unique identifier of the frame of reference

**Type**

*str*

**Return type**

*str*

**get\_channel\_identifier**(*channel\_index*)

Get identifier of a channel.

**Parameters**

**channel\_index** (*int*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute.

**Returns**

Channel identifier

**Return type**

*str*

**Raises**

**ValueError** – When no channel is found for *channel\_index*

**get\_channel\_index**(*channel\_identifier*)

Get index of a channel.

The nature of the channel is specific to the SOP Class for the image. For example, in case of DICOM VL Whole Slide Microscopy Image, a channel is an optical path and in case of a DICOM Segmentation, a channel is a segment.

**Parameters**

**channel\_identifier** (*str*) – Identifier of a channel

**Returns**

Zero-based index into channels along the direction defined by successive items of the corresponding attribute.

**Return type**

*int*

**Raises**

**ValueError** – When no channel is found for *channel\_identifier*

**get\_focal\_plane\_index**(*focal\_plane\_offset*)

Get index of a focal plane.

**Parameters**

**focal\_plane\_offset** (*float*) – Offset of the focal plane from the from the slide surface along the Z axis of the slide coordinate system in micrometers

**Returns**

Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values must be in the range [1, Total Pixel Matrix Focal Planes]

**Return type**

int

**Raises**

**ValueError** – When no focal plane is found for *focal\_plane\_offset*

**get\_focal\_plane\_offset**(*focal\_plane\_index*)

Get z offset in slide coordinate system of a focal plane.

**Parameters**

**focal\_plane\_index** (*int*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values must be in the range [0, Total Pixel Matrix Focal Planes).

**Returns**

Offset of the focal plane from the from the slide surface along the Z axis of the slide coordinate system in micrometers

**Return type**

float

**Raises**

**ValueError** – When no focal plane is found for *focal\_plane\_index*

**get\_image\_region**(*offset, size, channel\_index=0, focal\_plane\_index=0*)

Get image region.

**Parameters**

- **offset** (*Tuple[int, int]*) – Zero-based (row, column) indices in the range [0, Rows) and [0, Columns), respectively, that specify the offset of the image region in the total pixel matrix. The (0, 0) coordinate is located at the center of the topleft hand pixel in the total pixel matrix.
- **size** (*Tuple[int, int]*) – Rows and columns of the requested image region
- **channel\_index** (*int, optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute.
- **focal\_plane\_index** (*int, optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values must be in the range [0, Total Pixel Matrix Focal Planes)

**Returns**

Three-dimensional pixel array of shape (Rows, Columns, Samples per Pixel) for the requested image region

**Return type**

numpy.ndarray

**get\_pixel\_indices(*offset*)**

Get indices into total pixel matrix for a given slide position.

**Parameters**

**offset** (*Tuple*[*float*, *float*]) – Zero-based (x, y) offset in the slide coordinate system in millimeter

**Returns**

Zero-based (row, column) position in the total pixel matrix

**Return type**

*Tuple*[int, int]

---

**Note:** Pixel position may be negativ or extend beyond the size of the total pixel matrix if slide position at *offset* does fall into a region on the slide that was not imaged.

---

**get\_references(*sop\_class\_uid=None*)**

Get unique identifiers of referenced instances.

**Parameters**

**sop\_class\_uid** (*str*) – SOP Class UID of instances for which references should be obtained

**Returns**

Study, Series, and SOP Instance UID of each referenced image

**Return type**

List[*Tuple*[str, str, str]]

**get\_rotation()**

Get angle to rotate image such that it aligns with slide.

We want to align the image with the slide coordinate system such that the axes of the total pixel matrix are aligned with the X and Y axes of the slide coordinate system to ensure that spatial coordinates of graphic region of interest (ROI) annotations and are aligned with the source image region.

**Returns**

Counterclockwise angle of rotation

**Return type**

float

**get\_slide\_offset(*pixel\_indices*)**

Get slide coordinates for a given total pixel matrix position.

**Parameters**

**pixel\_indices** (*Tuple*[*int*, *int*]) – Zero-based (row, column) offset in the total pixel matrix

**Returns**

Zero-based (x, y) position on the slide in the slide coordinate system in millimeter

**Return type**

*Tuple*[float, float]

**get\_slide\_region**(*offset*, *size*, *channel\_index*=0, *focal\_plane\_index*=0)

Get slide region.

**Parameters**

- **offset** (*Tuple*[*float*, *float*]) – Zero-based (x, y) offset in the slide coordinate system in millimeter resolution. The (0.0, 0.0) coordinate is located at the origin of the slide (usually the slide corner).
- **size** (*Tuple*[*float*, *float*]) – Width and height of the requested slide region in millimeter unit along the X and Y axis of the slide coordinate system, respectively.
- **channel\_index** (*int*, *optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute.
- **focal\_plane\_index** (*int*, *optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values must be in the range [0, Total Pixel Matrix Focal Planes)

**Returns**

Three-dimensional pixel array of shape (Rows, Columns, Samples per Pixel) for the requested slide region

**Return type**

numpy.ndarray

---

**Note:** The slide coordinate system is defined for the upright standing slide such that the X axis corresponds to the short side of the slide and the Y axis corresponds to the long side of the slide. The rows of the returned pixel array are thus parallel to the X axis of the slide coordinate system and the columns parallel to the Y axis of the slide coordinate system.

---

**get\_total\_pixel\_matrix**(*channel\_index*=0, *focal\_plane\_index*=0)

Get total pixel matrix for a given optical path and focal plane.

**Parameters**

- **channel\_index** (*int*, *optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute.
- **focal\_plane\_index** (*int*, *optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values must be in the range [0, Total Pixel Matrix Focal Planes).

**Returns**

Total Pixel Matrix

**Return type**

*dicomslide.TotalPixelMatrix*

**map\_pixel\_indices\_to\_slide\_coordinates**(*pixel\_indices*)

Map pixel indices to slide coordinates.

**Parameters**

**pixel\_indices** (*numpy.ndarray*) – Zero-based (row, column) indices into the total pixel matrix of the image

**Returns**

Zero-based (x, y, z) coordinates in the slide coordinate system in millimeter

**Return type**

numpy.ndarray

**map\_slide\_coordinates\_to\_pixel\_indices**(*slide\_coordinates*)

Map slide coordinates to pixel indices.

**Parameters****slide\_coordinates** (*numpy.ndarray*) – Zero-based (x, y, z) coordinates in the slide coordinate system in millimeter**Returns**

Zero-based (row, column) indices into the total pixel matrix of the image

**Return type**

numpy.ndarray

**property metadata:** Dataset

Image metadata

**Type**

pydicom.dataset.Dataset

**Return type**

pydicom.dataset.Dataset

**property num\_channels:** int

Number of channels

**Type**

int

**Return type**

int

**property num\_focal\_planes:** int

Number of focal planes

**Type**

int

**Return type**

int

**property physical\_offset:** Tuple[float, float]

Offset of the total pixel matrix from the origin of the frame of reference along the X and Y axes of the slide coordinate system in millimeter

**Type**

Tuple[float, float]

**Return type**

typing.Tuple[float, float]

**property physical\_size:** Tuple[float, float]

Size of the total pixel matrix along the X and Y axes of the slide coordinate system in millimeter

**Type**

Tuple[float, float]

**Return type**

typing.Tuple[float, float]

**property size:** `Tuple[int, int]`

Number of total pixel matrix rows and columns

**Type**

`Tuple[int, int]`

**Return type**

`typing.Tuple[int, int]`

**class** `dicomslide.TotalPixelMatrix`(*client*, *image\_metadata*, *channel\_index*=0, *focal\_plane\_index*=0, *max\_frame\_cache\_size*=9, *correct\_color*=True)

Bases: `object`

Total Pixel Matrix.

The class exposes a NumPy-like interface to index into a total pixel matrix of a tiled image, where each tile is encoded as a separate frame. Instances of the class walk and quack like NumPy arrays and can be indexed accordingly. When the caller indexes instances of the class, the corresponding image frames are dynamically retrieved from a DICOM store and decoded.

A notable difference to NumPy array indexing is that a one-dimensional index returns an individual tile of the total pixel matrix (i.e., a 2D array) rather than an individual row of the total pixel matrix (i.e., a 1D array).

The caller can index instances of the class either using one-dimensional tile indices into the flattened list of tiles in the total pixel matrix to get one or more individual tiles or using three-dimensional pixel indices (rows, columns, and samples) into the total pixel matrix to get a continuous region of pixels spanning one or more tiles.

## Examples

```
>>> matrix = TotalPixelMatrix(...)
>>> print(matrix.dtype)
>>> print(matrix.ndim)
>>> print(matrix.shape)
>>> print(matrix.size)
>>> region = matrix[:256, 256:512, :]
>>> print(len(matrix))
>>> tile = matrix[0]
>>> tile = matrix[matrix.get_tile_index(2, 4)]
>>> tiles = matrix[[0, 1, 2, 5, 6, 7]]
>>> tiles = matrix[2:6]
```

**Warning:** The total pixel matrix may be very large and indexing the row or column dimension with `:` may consume a lot of time and memory.

Construct object.

### Parameters

- **client** (*dicomweb\_client.api.DICOMClient*) – DICOMweb client
- **image\_metadata** (*pydicom.dataset.Dataset*) – Metadata of a tiled DICOM image
- **channel\_index** (*int*, *optional*) – Zero-based index into channels along the direction defined by successive items of the appropriate DICOM attribute(s).



- **focal\_plane\_index** (*int*, *optional*) – Zero-based index into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values must be in the range [0, Total Pixel Matrix Focal Planes).
- **max\_frame\_cache\_size** (*int*, *optional*) – Maximum number of frames that should be cached to avoid repeated retrieval requests
- **correct\_color** (*bool*, *optional*) – Whether pixel values should be color corrected

**property dtype:** `dtype`

Data type

**Type**

`numpy.dtype`

**Return type**

`numpy.dtype`

**get\_tile\_bounding\_box**(*index*)

Get the bounding box of a tile.

**Parameters**

**index** (*int*) – Tile index

**Return type**

`typing.Tuple[typing.Tuple[int, int], typing.Tuple[int, int]]`

**Returns**

- **offset** (*Tuple[int, int]*) – Zero-based (row, column) pixel indices in the total pixel matrix
- **size** (*Tuple[int, int]*) – Height (rows) and width (columns) of the tile

**get\_tile\_grid\_position**(*index*)

Get position of a tile in the tile grid.

**Parameters**

**index** (*int*) – Zero-based index of the tile in the flattened total pixel matrix

**Returns**

Zero-based (row, column) index of a tile in the tile grid

**Return type**

`Tuple[int, int]`

**get\_tile\_index**(*position=None, frame\_number=None*)

Get index of a tile.

**Parameters**

- **position** (*Union[Tuple[int, int], None]*, *optional*) – Zero-based (row, column) index of a tile in the tile grid
- **frame\_number** (*Union[int, None]*, *optional*) – One-base number of the corresponding frame item in the Pixel Data element

**Returns**

Zero-based index of the tile in the flattened total pixel matrix

**Return type**

`int`

---

**Note:** Either *position* or *frame\_number* must be provided.

---

**get\_tile\_position**(*index*)

Get position of a tile.

**Parameters**

**index** (*int*) – Zero-based index of the tile in the flattened total pixel matrix

**Returns**

Zero-based (row, column) offset of a tile in the total pixel matrix

**Return type**

Tuple[int, int]

**property ndim: int**

Number of dimensions

**Type**

int

**Return type**

int

**property num\_tiles: int**

Number of tiles

**Type**

int

**Return type**

int

**property shape: Tuple[int, int, int]**

Rows, Columns, and Samples per Pixel

**Type**

Tuple[int, int, int]

**Return type**

typing.Tuple[int, int, int]

**property size: int**

Size (rows x columns x samples)

**Type**

int

**Return type**

int

**property tile\_grid\_positions: ndarray**

Two-dimensional array of integer values representing the grid positions of individual tiles in the tile grid

**Type**

numpy.ndarray

**Return type**

numpy.ndarray

**property tile\_grid\_shape: Tuple[int, int]**

Number of tiles along the column (top to bottom) and row (left to right) direction of the tile grid

**Type**

Tuple[int, int]

**Return type**

typing.Tuple[int, int]

**property tile\_positions: ndarray**

Two-dimensional array of integer values representing the positions of individual tiles in the total pixel matrix, i.e., the offsets from the (0, 0) origin of the total pixel matrix at the top lefthand pixel

**Type**

numpy.ndarray

**Return type**

numpy.ndarray

**property tile\_shape: Tuple[int, int, int]**

Number of pixel rows, pixel columns, and samples per pixel of an individual tile

**Type**

Tuple[int, int, int]

**Return type**

typing.Tuple[int, int, int]

**property tile\_size: int**

Size of an individual tile (rows x columns x samples)

**Type**

int

**Return type**

int

**class dicomslide.TotalPixelMatrixSampler**(*matrix, region\_dimensions, bounding\_box=None, tile\_grid\_positions=None, padding=0*)

Bases: object

Class for sampling regions of a total pixel matrix.

Regions are sampled from a regular 2D Cartesian grid, where each region has the same dimensions. Upon sampling, individual regions may optionally be padded at one or more borders using pixels from adjacent regions. Sampling can be constraint to a subset of the grid.

**Parameters**

- **matrix** (`dicomslide.TotalPixelMatrix`) – Total pixel matrix
- **region\_dimensions** (`Tuple[int, int]`) – Height (rows) and width (columns) of sampled regions
- **bounding\_box** (`Union[Tuple[Tuple[int, int], Tuple[int, int]], None]`, *optional*) – Bounding box of region of interest within total pixel matrix from which regions should be sampled
- **tile\_grid\_positions** (`Union[Sequence[Tuple[int, int]], numpy.ndarray, None]`, *optional*) – Grid position of tiles that intersect with the region of interest within the total pixel matrix from which regions should be sampled. Each grid position is a zero-based (row, column) index into the tile grid of the total pixel matrix.

- **padding** (*Union[int, Tuple[int, int], Tuple[int, int, int, int]], optional*) – Padding on each border of the sampled region using pixels from neighboring regions. If a single integer is provided, the value is used to pad all four borders with the same number of pixels. If a sequence of length 2 is provided, the two values are used to pad the left/right and top/bottom border, respectively. If a sequence of length 4 is provided, the four values are used to pad the left, top, right, and bottom borders respectively.

---

**Note:** If *bounding\_box* and *tile\_grid\_positions* are provided, *tile\_grid\_positions* are ignored.

---

**get\_region\_grid\_position**(*index*)

Get position of sampled region in the grid.

**Parameters**

**index** (*int*) – Zero-based index of the sampled region

**Returns**

Zero-based (row, column) grid position

**Return type**

Tuple[int, int]

**property matrix:** *TotalPixelMatrix*

Total pixel matrix

**Type**

*dicomslide.TotalPixelMatrix*

**Return type**

*dicomslide.matrix.TotalPixelMatrix*

**property padded\_region\_shape:** Tuple[int, int, int]

Number of pixel rows, pixel columns, and samples per pixel of sampled region with overlapping pixels from neighboring regions

**Type**

Tuple[int, int, int]

**Return type**

typing.Tuple[int, int, int]

**property padding:** Tuple[int, int, int, int]

Padding at the left, top, right, and bottom of each sampled region

**Type**

Tuple[int, int, int, int]

**Return type**

typing.Tuple[int, int, int, int]

**property region\_shape:** Tuple[int, int, int]

Number of pixel rows, pixel columns, and samples per pixel of a region

**Type**

Tuple[int, int, int]

**Return type**

typing.Tuple[int, int, int]

`dicomslide.assemble_total_pixel_matrix(tiles, tile_positions, total_pixel_matrix_rows, total_pixel_matrix_columns)`

Assemble a total pixel matrix from individual tiles.

#### Parameters

- **tiles** (*Sequence[numpy.ndarray]*) – Individual image tiles
- **tile\_positions** (*Union[Sequence[Tuple[int, int]], numpy.ndarray]*) – Zero-based (row, column) position of each tile in the total pixel matrix
- **total\_pixel\_matrix\_rows** (*int*) – Number of total rows
- **total\_pixel\_matrix\_columns** (*int*) – Number of total columns

#### Returns

Total pixel matrix

#### Return type

`numpy.ndarray`

`dicomslide.compute_frame_positions(image)`

Compute the positions of frames.

#### Parameters

**image** (*pydicom.Dataset*) – Metadata of a tiled image

#### Return type

`typing.Tuple[numpy.ndarray, numpy.ndarray, numpy.ndarray, numpy.ndarray]`

#### Returns

- **total\_pixel\_matrix\_positions** (*numpy.ndarray*) – Zero-based (row, column) offset of the center of the top lefthand corner pixel of each frame from the origin of the total pixel matrix in pixel unit. Values are unsigned integers in the range [0, Total Pixel Matrix Rows) and [0, Total Pixel Matrix Columns). The position of the top lefthand corner tile is (0, 0).
- **slide\_positions** (*numpy.ndarray*) – Zero-based (x, y, z) offset of the center of the top lefthand corner pixel of each frame from the origin of the slide coordinate system (frame of reference) in millimeter unit. Values are floating-point numbers in the range [-inf, inf].
- **channel\_indices** (*numpy.ndarray*) – Zero-based index for each frame into channels along the direction defined by successive items of the appropriate attribute. In case of a VL Whole Slide Microscopy Image, the attribute is the Optical Path Sequence, and in case of Segmentation, the attribute is the Segment Sequence.
- **focal\_plane\_indices** (*numpy.ndarray*) – Zero-based index for each frame into focal planes along depth direction from the glass slide towards the coverslip in the slide coordinate system specified by the Z Offset in Slide Coordinate System attribute. Values are integers in the range [0, Total Pixel Matrix Focal Planes).

`dicomslide.compute_image_center_position(image)`

Compute position of image center in slide coordinate system.

#### Parameters

**image** (*pydicom.dataset.Dataset*) – Metadata of DICOM VL Whole Slide Microscopy Image instance

#### Returns

(x, y, z) coordinates

#### Return type

`Tuple[float, float, float]`

`dicomslide.disassemble_total_pixel_matrix(total_pixel_matrix, tile_positions, rows, columns)`

Disassemble a total pixel matrix into individual tiles.

**Parameters**

- **total\_pixel\_matrix** (*numpy.ndarray*) – Total pixel matrix
- **tile\_positions** (*Union[Sequence[Tuple[int, int]], numpy.ndarray]*) – Zero-based (row, column) position of each tile in the total pixel matrix
- **rows** (*int*) – Number of rows per tile
- **columns** (*int*) – Number of columns per tile

**Returns**

Stacked image tiles

**Return type**

*numpy.ndarray*

`dicomslide.does_optical_path_item_match(item, identifier=None, description=None, illumination_wavelength=None)`

Check whether an optical path item matches.

**Parameters**

- **item** (*pydicom.Dataset*) – Item of the Optical Path Sequence
- **identifier** (*Union[str, None], optional*) – Optical path identifier
- **description** (*Union[str, None], optional*) – Optical path description
- **illumination\_wavelength** (*Union[float, None], optional*) – Illumination wavelength

**Returns**

Whether item matches

**Return type**

*bool*

`dicomslide.does_segment_item_match(item, number=None, label=None, property_category=None, property_type=None)`

Check whether a segment item matches.

**Parameters**

- **item** (*pydicom.Dataset*) – Item of the Segment Sequence
- **number** (*Union[int, None], optional*) – Segment number
- **label** (*Union[int, None], optional*) – Segment label
- **property\_category** (*Union[pydicom.sr.coding.Code, highdicom.sr.CodedConcept, None], optional*) – Segmented property category
- **property\_type** (*Union[pydicom.sr.coding.Code, highdicom.sr.CodedConcept, None], optional*) – Segmented property type

**Returns**

Whether item matches

**Return type**

*bool*

`dicomslide.does_specimen_description_item_match(item, specimen_stain=None)`

Check whether a specimen description item matches.

#### Parameters

- **item** (*pydicom.Dataset*) – Item of the Specimen Description Sequence
- **specimen\_stain** (*Union[pydicom.sr.coding.Code, highdicom.sr.CodedConcept, None]*, *optional*) – Specimen stain substance

#### Returns

Whether item matches

#### Return type

bool

`dicomslide.find_slides(client, study_instance_uid=None, patient_id=None, study_id=None, container_id=None, max_frame_cache_size=6, pyramid_tolerance=0.1, fail_on_error=True, include_derived=True, specimen_stains=None, optical_path_ids=None)`

Find slides.

#### Parameters

- **client** (*dicomweb\_client.api.DICOMClient*) – DICOMweb client
- **study\_instance\_uid** (*Union[str, None]*, *optional*) – DICOM Study Instance UID
- **patient\_id** (*Union[str, None]*, *optional*) – Patient identifier
- **study\_id** (*Union[str, None]*, *optional*) – Study identifier
- **container\_id** (*Union[str, None]*, *optional*) – Specimen container (slide) identifier
- **max\_frame\_cache\_size** (*int*, *optional*) – Maximum number of frames that should be cached per image instance to avoid repeated retrieval requests
- **pyramid\_tolerance** (*float*, *optional*) – Maximally tolerated distances between the centers of images at different pyramid levels in the slide coordinate system in millimeter unit
- **fail\_on\_error** (*bool*, *optional*) – Whether the function should raise an exception in case an error occurs. If False, slides will be skipped.
- **include\_derived** (*bool*, *optional*) – Whether derived images (DICOM Segmentation or DICOM Parametric Map instances) should be considered and included into slides
- **specimen\_stains** (*Union[Sequence[Union[pydicom.sr.Code, highdicom.sr.CodedConcept]], None]*) – Specimen stains for which corresponding slide microscopy images should be included in slides. Any image that does not contain one of the stains in the specimen description will be omitted.
- **optical\_path\_ids** (*Union[Sequence[str], None]*) – Identifiers of optical paths for which corresponding slide microscopy images should be included in slides. Any image that does not contain any of the specified optical paths will be omitted.

#### Returns

Digital slides

#### Return type

List[*dicomslide.Slide*]

`dicomslide.get_image_pixel_spacing(image)`

Get pixel spacing (spacing between pixels) of an image.

**Parameters**

**image** (*pydicom.dataset.Dataset*) – Metadata of a DICOM VL Whole Slide Microscopy Image instance derived image instance (e.g., DICOM Segmentation)

**Returns**

Pixel spacing

**Return type**

Tuple[float, float]

---

**Note:** It is assumed that pixels are square.

---

`dicomslide.get_image_size(image)`

Get size of an image.

**Parameters**

**image** (*pydicom.dataset.Dataset*) – Metadata of a DICOM VL Whole Slide Microscopy Image instance or a derived image instance (e.g., DICOM Segmentation)

**Returns**

Number of pixels in each total pixel matrix

**Return type**

int

`dicomslide.is_image(dataset)`

Determine whether a dataset is an image.

**Parameters**

**dataset** (*pydicom.dataset.Dataset*) – Dataset

**Returns**

Whether dataset is an image

**Return type**

bool

`dicomslide.is_label_image(dataset)`

Determine whether a dataset is a LABEL image.

**Parameters**

**dataset** (*pydicom.dataset.Dataset*) – Dataset

**Returns**

Whether dataset is a LABEL image

**Return type**

bool

`dicomslide.is_overview_image(dataset)`

Determine whether a dataset is an OVERVIEW image.

**Parameters**

**dataset** (*pydicom.dataset.Dataset*) – Dataset

**Returns**

Whether dataset is an OVERVIEW image

**Return type**

bool



`dicomslide.is_tiled_image(dataset)`

Determine whether a dataset is a tiled image.

**Parameters**

**dataset** (*pydicom.dataset.Dataset*) – Dataset

**Returns**

Whether dataset is a tiled image

**Return type**

bool

`dicomslide.is_volume_image(dataset)`

Determine whether a dataset is a VOLUME or THUMBNAIL image.

**Parameters**

**dataset** (*pydicom.dataset.Dataset*) – Dataset

**Returns**

Whether dataset is a VOLUME or THUMBNAIL image

**Return type**

bool

`dicomslide.select_image_at_magnification(collection, magnification, tolerance=None)`

Select an image from a collection at a desired magnification.

**Parameters**

- **collection** (*Sequence[pydicom.dataset.Dataset]*) – Metadata of DICOM VL Whole Slide Microscopy Image instances
- **magnification** (int) – Magnification level (corresponds roughly to object lens power of a microscope) of the image that should be selected. Note that an image with an exactly matching magnification may not exist. In this case, the nearest level will be chosen. Choices: {2, 4, 10, 20, 40}
- **tolerance** (*Union[float, None], optional*) – Difference between target magnification and closest available magnification in millimeter that can be tolerated.

**Returns**

Image closest to the desired magnification

**Return type**

*pydicom.dataset.Dataset*

**Raises**

**ValueError** – When argument *collection* is an empty sequence, when argument *magnification* does not match one of the available options, or when *tolerance* is exceeded.

`dicomslide.select_image_at_pixel_spacing(collection, pixel_spacing, tolerance=None)`

Select an image from a collection at a desired spatial pixel spacing.

**Parameters**

- **collection** (*Sequence[pydicom.dataset.Dataset]*) – Metadata of DICOM VL Whole Slide Microscopy Image instances
- **pixel\_spacing** (*Tuple[float, float]*) – Desired spacing between two pixels along the row and column direction of the image from top to bottom and left to right, respectively.
- **tolerance** (*Union[float, None], optional*) – Difference between target magnification and closest available magnification in millimeter that can be tolerated.

**Returns**

Image closest to the desired pixel spacing

**Return type**

pydicom.dataset.Dataset

**Raises**

**ValueError** – When argument *collection* is an empty sequence or when *tolerance* is exceeded.

---

**Note:** If multiple images with the same pixel spacing are contained in *collection*, the first matching image will be returned. It is the responsibility of the caller to filter the images beforehand if necessary.

---

`dicomslide.sort_images_by_pixel_spacing(collection)`

Sort images by pixel spacing in ascending order (lowest to highest).

**Parameters**

**collection** (*Sequence[pydicom.dataset.Dataset]*) – Metadata of DICOM VL Whole Slide Microscopy Image instances

**Returns**

Sorted metadata of DICOM VL Whole Slide Microscopy Image instances

**Return type**

List[pydicom.dataset.Dataset]

`dicomslide.sort_images_by_size(collection)`

Sort images by size in descending order (largest to smallest).

**Parameters**

**collection** (*Sequence[pydicom.dataset.Dataset]*) – Metadata of DICOM VL Whole Slide Microscopy Image instances

**Returns**

Sorted metadata of DICOM VL Whole Slide Microscopy Image instances

**Return type**

List[pydicom.dataset.Dataset]

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### d

`dicomslide`, [11](#)



## A

`assemble_total_pixel_matrix()` (in module *dicomslide*), 32  
`associated_images` (*dicomslide.OpenSlide* property), 12

## C

`channel_type` (*dicomslide.TiledImage* property), 23  
`ChannelTypes` (class in *dicomslide*), 11  
`close()` (*dicomslide.OpenSlide* method), 12  
`compute_frame_positions()` (in module *dicomslide*), 33  
`compute_image_center_position()` (in module *dicomslide*), 33

## D

`dicomslide`  
 module, 11  
`dimensions` (*dicomslide.OpenSlide* property), 12  
`disassemble_total_pixel_matrix()` (in module *dicomslide*), 34  
`does_optical_path_item_match()` (in module *dicomslide*), 34  
`does_segment_item_match()` (in module *dicomslide*), 34  
`does_specimen_description_item_match()` (in module *dicomslide*), 34  
`downsampling_factors` (*dicomslide.PyramidLevel* property), 14  
`downsampling_factors` (*dicomslide.Slide* property), 14  
`dtype` (*dicomslide.TotalPixelMatrix* property), 29

## F

`find_optical_paths()` (*dicomslide.Slide* method), 14  
`find_segments()` (*dicomslide.Slide* method), 15  
`find_slides()` (in module *dicomslide*), 35  
`frame_of_reference_uid` (*dicomslide.Slide* property), 15  
`frame_of_reference_uid` (*dicomslide.TiledImage* property), 23

## G

`get_best_level_for_downsample()` (*dicomslide.OpenSlide* method), 12  
`get_channel_identifier()` (*dicomslide.Slide* method), 15  
`get_channel_identifier()` (*dicomslide.TiledImage* method), 23  
`get_channel_index()` (*dicomslide.Slide* method), 16  
`get_channel_index()` (*dicomslide.TiledImage* method), 23  
`get_channel_type()` (*dicomslide.Slide* method), 16  
`get_focal_plane_index()` (*dicomslide.Slide* method), 16  
`get_focal_plane_index()` (*dicomslide.TiledImage* method), 24  
`get_focal_plane_offset()` (*dicomslide.Slide* method), 16  
`get_focal_plane_offset()` (*dicomslide.TiledImage* method), 24  
`get_image_pixel_spacing()` (in module *dicomslide*), 35  
`get_image_region()` (*dicomslide.Slide* method), 17  
`get_image_region()` (*dicomslide.TiledImage* method), 24  
`get_image_size()` (in module *dicomslide*), 36  
`get_pixel_indices()` (*dicomslide.Slide* method), 17  
`get_pixel_indices()` (*dicomslide.TiledImage* method), 25  
`get_references()` (*dicomslide.TiledImage* method), 25  
`get_region_grid_position()` (*dicomslide.TotalPixelMatrixSampler* method), 32  
`get_rotation()` (*dicomslide.TiledImage* method), 25  
`get_slide_offset()` (*dicomslide.Slide* method), 18  
`get_slide_offset()` (*dicomslide.TiledImage* method), 25  
`get_slide_region()` (*dicomslide.Slide* method), 18  
`get_slide_region()` (*dicomslide.TiledImage* method), 25  
`get_slide_region_for_annotation()` (*dicomslide.Slide* method), 19  
`get_thumbnail()` (*dicomslide.OpenSlide* method), 12

`get_tile_bounding_box()` (*dicomslide.TotalPixelMatrix* method), 29  
`get_tile_grid_position()` (*dicomslide.TotalPixelMatrix* method), 29  
`get_tile_index()` (*dicomslide.TotalPixelMatrix* method), 29  
`get_tile_position()` (*dicomslide.TotalPixelMatrix* method), 30  
`get_total_pixel_matrix()` (*dicomslide.TiledImage* method), 26  
`get_volume_images()` (*dicomslide.Slide* method), 19

## H

`has_pixels` (*dicomslide.PyramidLevel* property), 14

## I

`ImageFlavors` (class in *dicomslide*), 11  
`is_image()` (in module *dicomslide*), 36  
`is_label_image()` (in module *dicomslide*), 36  
`is_overview_image()` (in module *dicomslide*), 36  
`is_tiled_image()` (in module *dicomslide*), 36  
`is_volume_image()` (in module *dicomslide*), 37

## L

`LABEL` (*dicomslide.ImageFlavors* attribute), 11  
`label_images` (*dicomslide.Slide* property), 19  
`level_count` (*dicomslide.OpenSlide* property), 12  
`level_dimensions` (*dicomslide.OpenSlide* property), 12  
`level_downsamples` (*dicomslide.OpenSlide* property), 13

## M

`map_pixel_indices_to_slide_coordinates()` (*dicomslide.Slide* method), 20  
`map_pixel_indices_to_slide_coordinates()` (*dicomslide.TiledImage* method), 26  
`map_slide_coordinates_to_pixel_indices()` (*dicomslide.Slide* method), 20  
`map_slide_coordinates_to_pixel_indices()` (*dicomslide.TiledImage* method), 27  
`matrix` (*dicomslide.TotalPixelMatrixSampler* property), 32  
`metadata` (*dicomslide.TiledImage* property), 27  
module  
    *dicomslide*, 11

## N

`ndim` (*dicomslide.TotalPixelMatrix* property), 30  
`num_channels` (*dicomslide.Slide* property), 20  
`num_channels` (*dicomslide.TiledImage* property), 27  
`num_focal_planes` (*dicomslide.Slide* property), 21  
`num_focal_planes` (*dicomslide.TiledImage* property), 27

`num_levels` (*dicomslide.Slide* property), 21  
`num_tiles` (*dicomslide.TotalPixelMatrix* property), 30

## O

`OpenSlide` (class in *dicomslide*), 11  
`OPTICAL_PATH` (*dicomslide.ChannelTypes* attribute), 11  
`OVERVIEW` (*dicomslide.ImageFlavors* attribute), 11  
`overview_images` (*dicomslide.Slide* property), 21

## P

`padded_region_shape` (*dicomslide.TotalPixelMatrixSampler* property), 32  
`padding` (*dicomslide.TotalPixelMatrixSampler* property), 32  
`PARAMETER` (*dicomslide.ChannelTypes* attribute), 11  
`physical_offset` (*dicomslide.Slide* property), 21  
`physical_offset` (*dicomslide.TiledImage* property), 27  
`physical_size` (*dicomslide.Slide* property), 21  
`physical_size` (*dicomslide.TiledImage* property), 27  
`pixel_spacing` (*dicomslide.PyramidLevel* property), 14  
`pixel_spacings` (*dicomslide.Slide* property), 21  
`properties` (*dicomslide.OpenSlide* property), 13  
`Pyramid` (class in *dicomslide*), 13  
`PyramidLevel` (class in *dicomslide*), 13

## R

`read_region()` (*dicomslide.OpenSlide* method), 13  
`region_shape` (*dicomslide.TotalPixelMatrixSampler* property), 32

## S

`SEGMENT` (*dicomslide.ChannelTypes* attribute), 11  
`select_image_at_magnification()` (in module *dicomslide*), 37  
`select_image_at_pixel_spacing()` (in module *dicomslide*), 37  
`shape` (*dicomslide.TotalPixelMatrix* property), 30  
`size` (*dicomslide.Slide* property), 22  
`size` (*dicomslide.TiledImage* property), 27  
`size` (*dicomslide.TotalPixelMatrix* property), 30  
`Slide` (class in *dicomslide*), 14  
`sort_images_by_pixel_spacing()` (in module *dicomslide*), 38  
`sort_images_by_size()` (in module *dicomslide*), 38

## T

`THUMBNAIL` (*dicomslide.ImageFlavors* attribute), 11  
`tile_grid_positions` (*dicomslide.TotalPixelMatrix* property), 30  
`tile_grid_shape` (*dicomslide.TotalPixelMatrix* property), 30



`tile_positions` (*dicomslide.TotalPixelMatrix* property), [31](#)  
`tile_shape` (*dicomslide.TotalPixelMatrix* property), [31](#)  
`tile_size` (*dicomslide.TotalPixelMatrix* property), [31](#)  
`TiledImage` (class in *dicomslide*), [22](#)  
`total_pixel_matrix_dimensions` (*dicomslide.PyramidLevel* property), [14](#)  
`total_pixel_matrix_dimensions` (*dicomslide.Slide* property), [22](#)  
`TotalPixelMatrix` (class in *dicomslide*), [28](#)  
`TotalPixelMatrixSampler` (class in *dicomslide*), [31](#)

## V

`VOLUME` (*dicomslide.ImageFlavors* attribute), [11](#)