

---

**dicom-wsi**

*Release 0.1.0*

Sep 29, 2020



---

## Contents:

---

<b>1</b>	<b>dicom_wsi</b>	<b>1</b>
<b>2</b>	<b>Usage</b>	<b>3</b>
2.1	Features . . . . .	3
2.2	TODO . . . . .	3
2.3	Credits . . . . .	3
<b>3</b>	<b>Installation</b>	<b>5</b>
3.1	With Conda (Preferred) . . . . .	5
3.2	Stable release . . . . .	6
3.3	From sources . . . . .	6
3.4	With Docker . . . . .	6
3.5	Development with PyCharm . . . . .	6
<b>4</b>	<b>TL;DR</b>	<b>7</b>
<b>5</b>	<b>Getting Started</b>	<b>9</b>
5.1	Create a YAML file . . . . .	9
5.2	Without a YAML file . . . . .	10
5.3	Using in existing code . . . . .	10
5.4	Sample RUN . . . . .	10
<b>6</b>	<b>Examples</b>	<b>11</b>
6.1	Other functions . . . . .	12
<b>7</b>	<b>Working with iSyntax files</b>	<b>13</b>
<b>8</b>	<b>Annotations</b>	<b>15</b>
8.1	Getting some sample XML data . . . . .	16
8.2	Inserting into the DICOM file . . . . .	18
<b>9</b>	<b>Extracting Annotations &amp; Images from DICOM Files</b>	<b>21</b>
9.1	Extract Annotations from DICOM . . . . .	21
9.2	Extract Images from Dicom . . . . .	21
<b>10</b>	<b>Code</b>	<b>23</b>
10.1	Base Attributes . . . . .	23
10.2	Character Validation . . . . .	23

10.3	Input Validation . . . . .	23
10.4	Mapping Functions . . . . .	24
10.5	Whole Slide Image Parsing . . . . .	24
10.6	Down-sampling the WSI . . . . .	24
10.7	Converting Pixels to Positions . . . . .	24
10.8	Adding in Sequence Data . . . . .	26
10.9	Adding Functional Groups Data . . . . .	26
10.10	Other helper utilities . . . . .	26
<b>11</b>	<b>Contributing</b>	<b>29</b>
11.1	Types of Contributions . . . . .	29
11.2	Get Started! . . . . .	30
11.3	Pull Request Guidelines . . . . .	31
11.4	Tips . . . . .	31
11.5	Deploying . . . . .	31
<b>12</b>	<b>Credits</b>	<b>33</b>
12.1	Development Lead . . . . .	33
12.2	Contributors . . . . .	33
<b>13</b>	<b>History</b>	<b>35</b>
13.1	0.1.0 (2019-10-02) . . . . .	35
<b>14</b>	<b>Indices and tables</b>	<b>37</b>
	<b>Python Module Index</b>	<b>39</b>
	<b>Index</b>	<b>41</b>

# CHAPTER 1

---

dicom\_wsi

---

---

Package for converting whole slide image files to DICOM.

- Free software: MIT license
- Documentation: <https://dicom-wsi.readthedocs.io>.



First, you need to install `dicom_wsi` and its dependencies. See this [link](#) for details.

To use `dicom-wsi`:

```
python cli.py cli.py -w <WSI File path> -o <OutputDirectory> -p <output file prefix> -  
→y yaml/base.yaml
```

That's it! Most of the time you wan't need to change anything. But if you do, please see the example [yaml](#) file.

## 2.1 Features

- Validate DICOM elements using [pydicom](#)
- Output format DICOM formatted files (vetted with [dciodvfy](#))

## 2.2 TODO

- Find out how to determine what *FileMetaInformationGroupLength* should be

## 2.3 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.





---

**Important:** The `libvips` and `OpenSlide` packages are required, but not available on PyPi. You will need to make sure your environment has these packages available.

---

### 3.1 With Conda (Preferred)

The preferred method uses `miniconda` because there are several necessary modules that cannot be installed with `pip`. To install `miniconda`:

```
$ sudo apt-get update
$ wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh -O_
↳miniconda.sh;
$ bash miniconda.sh -b -p $HOME/miniconda
$ source "$HOME/miniconda/etc/profile.d/conda.sh"
$ hash -r
$ conda config --set always_yes yes --set changeps1 no
$ conda update -q conda
```

Once `conda` is installed, then you can install `dicom_wsi`:

```
$ conda config --add channels bioconda
$ conda config --add channels conda-forge
$ conda create -q -n test-environment python pyvips openslide openjpeg libtiff
$ conda activate test-environment
$ pip install -U -r requirements_dev.txt
```

If using `iSyntax` files, you also need `libtiff` <http://download.osgeo.org/libtiff/tiff-4.1.0.zip>

## 3.2 Stable release

To install dicom-wsi, run this command in your terminal (assuming you already have the non-pip installed libraries):

```
$ pip install dicom_wsi
```

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

## 3.3 From sources

The sources for dicom-wsi can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/Steven-N-Hart/dicom_wsi
```

Or download the [tarball](#):

```
$ curl -OJL https://github.com/Steven-N-Hart/dicom_wsi/tarball/master
```

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

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

## 3.4 With Docker

You can also build a container using Docker:

```
$ docker build -t stevenhart/dicom_wsi .
```

## 3.5 Development with PyCharm

If you are going to do some development work with PyCharm, you will need to copy the binary files into your venv.

## CHAPTER 4

---

TL;DR

---

To use dicom-wsi in a project:

```
python cli.py cli.py -w <WSI File path> -o <OutputDirectory> -p <output file prefix> -  
→y yaml/base.yaml
```

That's it!



Before you can run *dicom\_wsi*, you first need to gather some input data necessary for creating a valid DICOM file. There are two ways to do this: 1.) Create a YAML file or 2.) Build a dictionary yourself.

## 5.1 Create a YAML file

The *YAML* file contains two sections: *General* and *BaseAttributes*. The *General* section contains a fixed number of required fields that are used by *dicom\_wsi*, whereas the *BaseAttributes* are restricted key words from the DICOM standard. These are cross-referenced with the *pydicom* dictionary.

### 5.1.1 *General* section

#### Definitions

Term	Definition
<i>WSIFile</i>	Path to whole slide image file
<i>OutFilePrefix</i>	What prefix to use when saving the DICOM output file
<i>NumberOfLevels</i>	How many levels should be extracted (in powers of 2)
<i>OrgUIDRoot</i>	Your organizations UID root prefix
<i>FrameSize</i>	How many pixels should be used for DICOM frames
<i>MaxFrames</i>	Number of frames allowed before writing to a new file

### 5.1.2 *BaseAttributes* section

Most of the terms in this section are defined in the DICOM standard. Many will not need to be changed, but some always will. Below, I highlight those terms that will likely need to be manually set. **Definitions**

Term	Definition
<i>PatientName</i>	LastName^FirstName
<i>PatientBirthDate</i>	Date format (i.e. 20000101)
<i>PatientSex</i>	M: Male, F: Female, O: Other
<i>PatientID</i>	Unique identifier for the patient
<i>ReferringPhysicianName</i>	LastName^FirstName
<i>StudyDate</i>	Date format
<i>StudyID</i>	Human readable study name

A full example can be found in *yaml/base.yaml*.

## 5.2 Without a YAML file

While a *YAML* file is recommended, you don't actually need one. You could choose to make the dictionary yourself. The dictionary has two nested components, *General* and *BaseAttributes*, each of which has the elements defined in *yaml/base.yaml*.

## 5.3 Using in existing code

To use dicom-wsi in a project:

```
from yaml import load, BaseLoader
from dicom_wsi import create_dicom
from parse_wsi import get_wsi

# Define your YAML file
my_yaml = '/path/to/yaml'
# Load your YAML file
cfg = load(open(my_yaml), Loader=BaseLoader)
# Read the WSI, updating the config with information contained in the slide
cfg, wsi = get_wsi(cfg)
# Create DICOM files
create_dicom(cfg)
```

## 5.4 Sample RUN

This Step will download the sample svx file `python ./tests/__init__.py`

This is sample execution `python cli.py -y ./tests/testfiles/base.yaml`

## CHAPTER 6

---

### Examples

---

Here is an end to end example

Clone the git repo

```
$ git clone https://github.com/Steven-N-Hart/dicom_wsi.git
$ cd dicom_wsi
$ mkdir example
$ cd example
```

Install the required packages as described [here](#)

Downloading the svcs file

```
$ wget http://openslide.cs.cmu.edu/download/openslide-testdata/Aperio/CMU-1-JP2K-
↪33005.svs
```

getting the annotations file

```
$ cp ../tests/CMU-1-JP2K-33005.xml .
```

Getting the input yaml file to generate the dicom file. Modifying values for params 'WSI-File', 'OutFilePrefix', 'Annotations'.

Annotations are optional, if you want to skip annotations, then remove 'Annotations' param in the base.yaml file. Below command will replace the paths for params 'WSIFile', 'OutFilePrefix', 'Annotations' to current directory

```
$ cat ../dicom_wsi/yaml/base.yaml |sed -e 's/tests\\///g'|sed -e 's/.\\///g' > base.yaml
```

Running the dicom\_wsi tool & generating the dicom files

```
$ python ../dicom_wsi/cli.py -y base.yaml
```

Following dicom files will be generated (Multiple dicom files for multiple levels).

```
$ ls output.*.dcm
```

output.0-10.dcm  
output.0-2.dcm  
output.0-6.dcm  
output.1-1.dcm  
output.3-0.dcm  
output.0-11.dcm  
output.0-3.dcm  
output.0-7.dcm  
output.1-2.dcm  
output.4-0.dcm  
output.0-12.dcm  
output.0-4.dcm  
output.0-8.dcm  
output.1-3.dcm  
output.5-0.dcm  
output.0-1.dcm  
output.0-5.dcm  
output.0-9.dcm  
output.2-0.dcm  
output.6-0.dcm

Optional: Validating the generated dicom files.

Download this tool [dciodvfy](#) to validate the generated dicom files

## 6.1 Other functions

Extracting Annotations from Dicom file to a python dictionary(Here i'm running it on only one level Dicom file)

```
$ python ../dicom_wsi/mods/extract_annotations.py -D output.2-0.dcm
```

Extracting images from Dicom file

```
$ python ../dicom_wsi/mods/extract_image_patches.py -D output.2-0.dcm -d output_  
↪ images
```



---

## Working with iSyntax files

---

If you are like me and are stuck with iSyntax files, then they first need to be converted to TIFF. The following script will parse the isyntax file to make a config file and a BigTiff file.

First, change directories so you are in the isyntax directory of this repo , and download an example iSyntax file from the [Phillips Website](#).

```
$ cd isyntax
$ curl -o ex1.isyntax https://www.openpathology.philips.com/wp-content/uploads/
↳AnimalSlides/1.isyntax
```

Next, get the Phillips SDK. You need to create a log in first.

You will need to follow the installation instructions for your specific operating system. Once you have it installed, open a python terminal and run:

```
import pixelengine
```

If you get *ModuleNotFoundError: No module named 'pixelengine'* then you do not have this installed properly. Ask Phillips tech support for help.

Once you have the Phillips SDK installed, you can run the conversion script.

```
$ python isyntax_to_tiff.py --input 1.isyntax --tif BIGTIFF --sparse 0 --startlevel 0
```

This will create a file called *ex1\_BIG\_sparse.tiff*.

Now you can create the configuration file and proceed as normal.



## CHAPTER 8

---

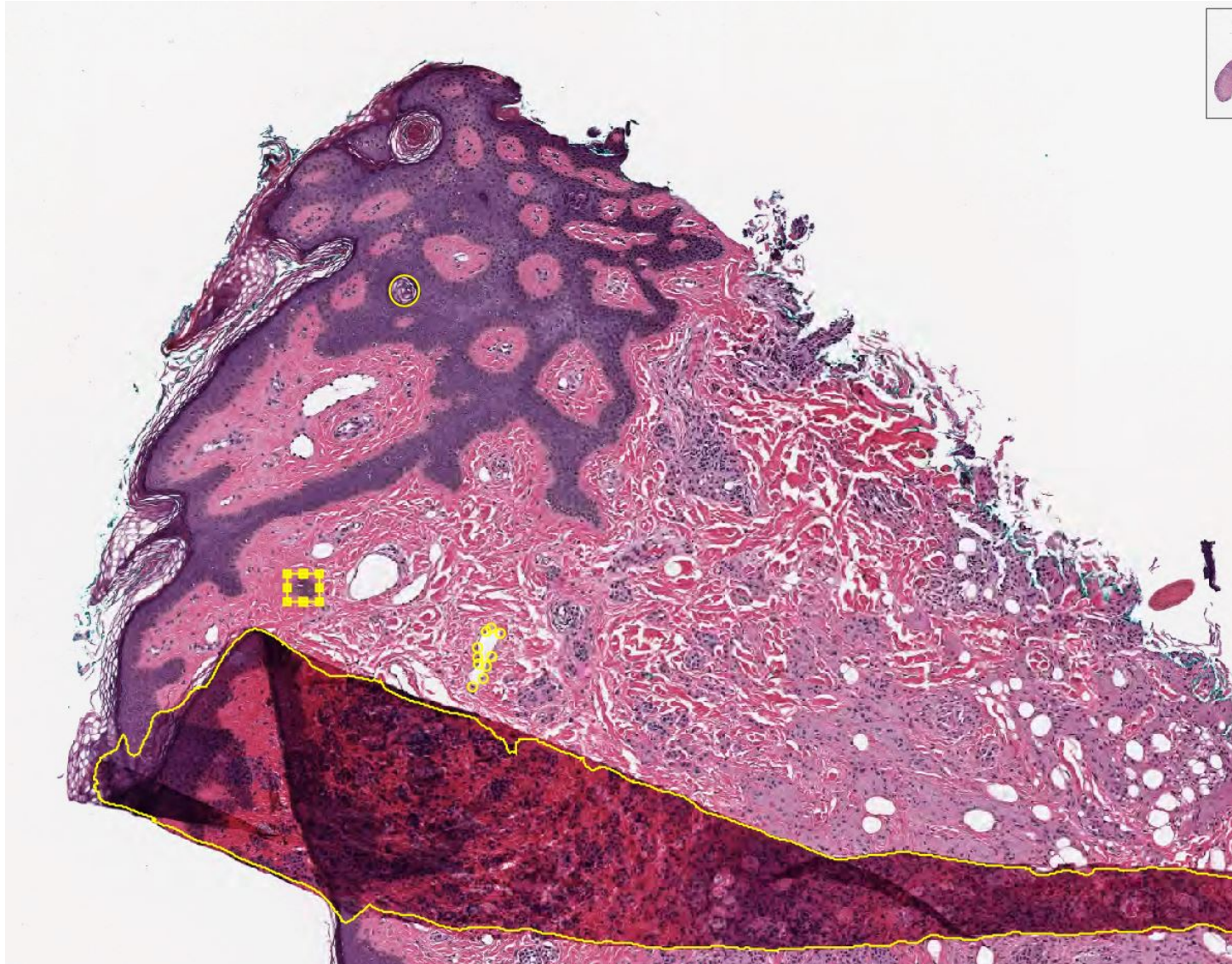
### Annotations

---

Given an XML file of annotations, extract the data into the appropriate DICOM element.

If you have Annotations in the XML file structure listed below, and you want to include them in your DICOM file, then all you need to do is to add the Annotations key and the associated file path to the *General* attribute in the YAML file.

**General:** Annotations: '/path/to/xmlFile'



## 8.1 Getting some sample XML data

We have prepared some different annotations on the [Aperio](#) example CMU-1-JP2K-33005.svs. These examples were drawn using [QuPath](#) and extracted with [this](#) script. The annotations have no physiological relevance, only to show how the different data types can be stored inside DICOM. Each of the yellow markups in the image above describe one of the following data types:

- Points
- Rectangle
- Area
- Ellipse

All of the data types should be encoded in an XML tree that looks like the following:

```

<Annotations>
  <Annotation>
    <Regions>
      <Region Text="null" GeoShape="Points">
        ...
      </Region>
    </Regions>
  </Annotation>
</Annotations>

```

The important characteristics here are *Text="null"* and *GeoShape="Points"*. These define a human readable label(*Text*), and a data type (*GeoShape*). Each will be discussed below.

### 8.1.1 Points

The simplest annotation type is the point. It defines as specific x and y coordinate (pixel). Each entry in this group represents an independent data point. If there are points that have different meanings (e.g. mitosis vs lymphocyte), then they should be grouped in a different section with a different *Text* value.

```

<Region Id="1" Type="0" Text="null" GeoShape="Points" Zoom="0.042148" Selected="0"
↪ImageLocation="" ImageFocus="0" Length="74565.8" Area="213363186.2" LengthMicrons=
↪"18798.0" AreaMicrons="13560170.4" NegativeROA="0" InputRegionId="0" Analyze="1"
↪DisplayId="1">
  <Attributes/>
  <Vertices>
    <Vertex X="37279.312500" Y="4662.189453"/>
    <Vertex X="37319.550781" Y="4588.894043"/>
    <Vertex X="..." Y="..."/>
  </Vertices>
</Region>

```

### 8.1.2 Rectangle

The next simplest annotation is a rectangle, or bounding box. These annotations define an area that contains an object of interest. They require four different points to describe the boundaries of the x and y corners.

```

<Region Id="2" Type="2" Text="Necrosis" GeoShape="Rectangle" Zoom="0.042148" Selected=
↪"0" ImageLocation="" ImageFocus="0" Length="74565.8" Area="213363186.2"
↪LengthMicrons="18798.0" AreaMicrons="13560170.4" NegativeROA="0" InputRegionId="0"
↪Analyze="1" DisplayId="1">
  <Attributes/>
  <Vertices>
    <Vertex X="36406.388563" Y="4324.243648"/>
    <Vertex X="36554.076020" Y="4324.243648"/>
    <Vertex X="36554.076020" Y="4452.625555"/>
    <Vertex X="36406.388563" Y="4452.625555"/>
  </Vertices>
</Region>

```

### 8.1.3 Area

An area annotations structured identically to the bounding box, except that there can be any number of x,y coordinate pairs. This is the annotation typically used for image segmentation.

```

<Region Id="3" Type="1" Text="Fold" GeoShape="Area" Zoom="0.042148" Selected="0"
↳ImageLocation="" ImageFocus="0" Length="74565.8" Area="213363186.2" LengthMicrons=
↳"18798.0" AreaMicrons="13560170.4" NegativeROA="0" InputRegionId="0" Analyze="1"
↳DisplayId="1">
  <Attributes/>
  <Vertices>
    <Vertex X="36382.175781" Y="4644.585938"/>
    <Vertex X="36389.238281" Y="4651.647949"/>
    ...
    <Vertex X="36262.121094" Y="4573.966309"/>
    <Vertex X="36255.058594" Y="4573.966309"/>
  </Vertices>
</Region>

```

### 8.1.4 Ellipse

Ellipses are just circular annotations. They have the same structure as Rectangles, but rather than being connected by straight lines in an image viewer, they will instead be connected with curved lines.

```

<Region Id="4" Type="0" Text="null" GeoShape="Ellipse" Zoom="0.042148" Selected="0"
↳ImageLocation="" ImageFocus="0" Length="74565.8" Area="213363186.2" LengthMicrons=
↳"18798.0" AreaMicrons="13560170.4" NegativeROA="0" InputRegionId="0" Analyze="1"
↳DisplayId="1">
  <Attributes/>
  <Vertices>
    <Vertex X="36943.806573" Y="2957.558623"/>
    <Vertex X="37011.368077" Y="3027.752393"/>
    <Vertex X="36943.806573" Y="3097.946164"/>
    <Vertex X="36876.245069" Y="3027.752393"/>
  </Vertices>
</Region>

```

## 8.2 Inserting into the DICOM file

This process assumes you have a `pydicom` object called `ds`. Let's go ahead and build out the base for our annotations.

```

from pydicom.sequence import Sequence
from pydicom.dataset import Dataset

ds = ... # Stuff to create DICOM file
dsDisplayedArea = Dataset()
dsDisplayedArea.PresentationSizeMode = 'TRUE SIZE'
ds.DisplayedAreaSelectionSequence = Sequence([dsDisplayedArea])
ds.GraphicAnnotationSequence = Sequence([])
ds.GraphicAnnotationSequence[0].ReferencedImageSequence = Sequence([])
ds.GraphicAnnotationSequence[0].ReferencedImageSequence[0].GraphicObjectSequence =
↳Sequence([])

```

Determine what type of annotation element is needed:

1. Rectangle

```

# Graphics on the first referenced image
GraphicObjectSequence = Dataset()
GraphicObjectSequence.BoundingBoxTopLeftHandCorner = [36406.388563, 4452.625555]
GraphicObjectSequence.BoundingBoxBottomRightHandCorner = [36554.076020, 4324.243648]
↪# bottom right coordinates of bounding box [max_x, min_y]
GraphicObjectSequence.BoundingBoxAnnotationUnits = 'PIXEL' # unit of coordinates
GraphicObjectSequence.BoundingBoxHorizontalJustification = 'LEFT'
GraphicObjectSequence.UnformattedTextValue = 'Necrosis' # Text="Necrosis"
GraphicObjectSequence.GraphicGroupID = '2' # Id="2"
gos = Sequence([GraphicObjectSequence])
ds.GraphicAnnotationSequence[0].ReferencedImageSequence[0].GraphicObjectSequence.
↪append(gos)
del GraphicObjectSequence
del gos

```

## 2. Points

```

GraphicObjectSequence = Dataset()
GraphicObjectSequence.GraphicType = "POINT"
GraphicObjectSequence.NumberofGraphicPoints = 4 # how many points where saved in_
↪this domain, validate data is complete
GraphicObjectSequence.GraphicData = [37279.312500, 4662.189453, 37319.550781, 4588.
↪894043, ..., ...] # x,y coordinates of points [x0, y0, x1, y1 ...]
GraphicObjectSequence.GraphicAnnotationUnits = 'PIXEL' # unit of coordinates
GraphicObjectSequence.GraphicGroupID = '1' # Id="1" Type="0" Text="null"
gos = Sequence([GraphicObjectSequence])
ds.GraphicAnnotationSequence[0].ReferencedImageSequence[0].GraphicObjectSequence.
↪append(gos)
del GraphicObjectSequence
del gos

```

## 3. Area

```

GraphicObjectSequence = Dataset()
GraphicObjectSequence.GraphicType = "POLYLINE" # add polyline
GraphicObjectSequence.NumberofGraphicPoints = 4 # how many points where saved in this_
↪domain
GraphicObjectSequence.GraphicData = [36382.175781, 4644.585938, 36389.238281, 4651.
↪647949, ..., ...]
GraphicObjectSequence.GraphicAnnotationUnits = 'PIXEL' # unit of coordinates
GraphicObjectSequence.GraphicGroupID = 3 # Annotation Label ID: 2
gos = Sequence([GraphicObjectSequence])
ds.GraphicAnnotationSequence[0].ReferencedImageSequence[0].GraphicObjectSequence.
↪append(gos)
del GraphicObjectSequence
del gos

```

## 4. Ellipse

```

GraphicObjectSequence = Dataset()
GraphicObjectSequence.GraphicType = "ELLIPSE"
GraphicObjectSequence.NumberofGraphicPoints = 4 # how many points where saved in this_
↪domain
GraphicObjectSequence.GraphicData = [36943.806573, 2957.558623, 37011.368077, 3027.
↪752393, 36943.806573, 3097.946164, 36876.245069, 3027.752393]
GraphicObjectSequence.GraphicAnnotationUnits = 'PIXEL' # unit of coordinates
GraphicObjectSequence.GraphicGroupID = 3 # Annotation Label ID: 2

```

(continues on next page)

(continued from previous page)

```
gos = Sequence([GraphicObjectSequence ])
ds.GraphicAnnotationSequence[0].ReferencedImageSequence[0].GraphicObjectSequence.
→append(gos)
del GraphicObjectSequence
del gos
```



---

## Extracting Annotations & Images from DICOM Files

---

### 9.1 Extract Annotations from DICOM

The following script will parse the DICOM file to extract annotations and returns the annotations in a python dictionary object.

```
$ python dicom_wsi/mods/extract_annotations.py -D <Path to DICOM file>
```

or from inside a python script:

```
from dicom_wsi.mods.extract_annotations import extract_ann_dicom
d = extract_ann_dicom('tests/output.6-0.dcm') # Change this to your dicom file
```

### 9.2 Extract Images from Dicom

The following script will parse the DICOM file to extract images and write them to a specified directory.

```
$ python ./dicom_wsi/mods/extract_image_patches.py -D <Path to Dicom file> -d <Path_
↳to output directory>
```



## 10.1 Base Attributes

`dicom_wsi.mods.base_attributes.build_base(cfg, instance=1)`

Creates the initial file and configuration, as well as sets defaults and 1st level data.

### Parameters

- **cfg** – dictionary of input values
- **instance** – ‘BaseAttributes’ in case we wanted to use this function later

**Returns** DICOM and new config files

## 10.2 Character Validation

## 10.3 Input Validation

`dicom_wsi.mods.input_validation.validate_cfg(cfg)`

Validate the dictionary to make sure all the required elements are present :param cfg: a dictionary of values, separated by module name :return: 0

## 10.4 Mapping Functions

## 10.5 Whole Slide Image Parsing

## 10.6 Down-sampling the WSI

`dicom_wsi.mods.pixel_data_conversion.resize_wsi_image` (*wsi=None, series\_downsample=0*) *se-*

reshape the pyvips object to the desired size based on series downsample

### Parameters

- **wsi** – PyVIPS Object
- **series\_downsample** – How many times to downsample

**Returns** byte string to use as the pixel array

## 10.7 Converting Pixels to Positions

`dicom_wsi.mods.pixel_to_slide_conversions.add_imgdata` (*imlist, ds, tile\_size, compression\_type, compression\_quality*)

Calls the compression algorithm and add to DICOM object

### Parameters

- **imlist** – list of images to compress
- **ds** – DICOM object
- **tile\_size** – how large the image tiles should be
- **compression\_type** – name of compression scheme
- **compression\_quality** – quality of compression

**Returns** DICOM object with pixel data

`dicom_wsi.mods.pixel_to_slide_conversions.add_per_frame_functional_groups_sequence` (*img=None, ds=None, cfg=None, tile\_size=5, series\_downs*)

Calculate the PerFrame Functional Groups

### Parameters

- **img** – VIPS image object
- **ds** – DICOM Object
- **cfg** – Config dictionary
- **tile\_size** – how big should each sub image be?
- **series\_downsample** – Factor to translate between inches and pixels

**Returns** None

dicom\_wsi.mods.pixel\_to\_slide\_conversions.**compute\_slide\_offsets\_from\_pixel\_data** (*ds=None, row=None, col=None, series\_downsample*)

Calculate the x and y coordinate in slide space

**Parameters**

- **ds** – some sort of DICOM object
- **row** – tile row number [Dimension Index Values (0020,9157)]
- **col** – tile column number [Dimension Index Values (0020,9157)]
- **series\_downsample** – number that indicates how many divisions to apply (1 means no downsample, i.e. level 0)

**Returns** x (0040,072a), y (0040,073a), and z (0040,074a) offsets

dicom\_wsi.mods.pixel\_to\_slide\_conversions.**define\_plane\_position\_slide\_sequence** (*x, y, z, x\_tile, y\_tile, x\_pos, y\_pos*)

Build up the sequence position structure for the coordinates

**Parameters**

- **x** – offset position on slide
- **y** – offset position on slide
- **z** – offset position on slide (usually 1)
- **x\_tile** – tile position number
- **y\_tile** – tile position number
- **x\_pos** – pixel position
- **y\_pos** – pixel position

**Returns** a dataset value to be appended to the PerFrameFunctionalGroupsSequence

dicom\_wsi.mods.pixel\_to\_slide\_conversions.**ensure\_even\_image** (*tmp, tile\_size*)  
Images must be an even before being compressed

**Parameters**

- **tmp** – numpy array of image values
- **tile\_size** – desired output size

**Returns** an even numbered shape for the numpy array

dicom\_wsi.mods.pixel\_to\_slide\_conversions.**generate\_xy\_tiles** (*x\_max, y\_max, tile\_size=500*)

Iterate through the slide with a step size of *tile\_size*

**Parameters**

- **x\_max** –
- **y\_max** –

- **tile\_size** –

**Returns** ( tile position on the x plane, tile position on the y plane, index of the tile on the x plane, index of the tile on the y plane)

## 10.8 Adding in Sequence Data

`dicom_wsi.mods.sequence_attributes.build_sequences` (*dcm*)  
all values are hard coded to ensure they are present in the final file

## 10.9 Adding Functional Groups Data

`dicom_wsi.mods.shared_functional_groups.build_functional_groups` (*dcm, cfg*)  
all values are hard coded to ensure they are present in the final file

## 10.10 Other helper utilities

`dicom_wsi.mods.utils.add_data` (*ds, k, v, cfg, dict\_element=None*)  
Ensures the right kind of data is insreted into DICOM object (e.g. datetime, time, date, etc)

### Parameters

- **ds** – DICOM dataset
- **k** – DICOM keyword
- **v** – Value to set keyword
- **cfg** – config dict
- **dict\_element** – which part of processing is this coming from

### Returns

`dicom_wsi.mods.utils.get_all_keys` (*d, prefix=False*)  
Return a list of all keys (even nested ones from dict)

### Parameters

- **d** – nested dictionary
- **prefix** – Whether or not to provide dot notation to keep the nesting information

**Returns** list of keys

`dicom_wsi.mods.utils.get_info_from_keyword` (*kw*)  
Get Value data from Keywords

**Parameters** **kw** – KeyWord

**Returns** VR, VM, NAME, ISRETIRED, KW

`pydicom.datadict.get_entry`('StudyDate') ('DA', '1', 'Study Date', '', 'StudyDate')

`dicom_wsi.mods.utils.make_date` (*k, date\_var, cfg, dict\_element='BaseAttributes'*)  
Need to make sure it return the format YYYYMMDD, or return a new one

### Parameters

- **k** – key
- **date\_var** – value
- **cfg** – config file
- **dict\_element** – which section/module the attributes belong to

**Returns** updated config dictionary

`dicom_wsi.mods.utils.make_datetime(k, datetime_var, cfg, dict_element='BaseAttributes')`

Need to make sure it return the format YYYYMMDDHHMMSS.FFFFFFFF, or return a new one

**Parameters**

- **k** – key
- **datetime\_var** – value
- **cfg** – config file
- **dict\_element** – which section/module the attributes belong to

**Returns** updated config dictionary

`dicom_wsi.mods.utils.make_time(k, time_var, cfg, dict_element='BaseAttributes')`

Need to make sure it return the format HHMMSS.FFFFFFFF, or return a new one

**Parameters**

- **k** – key
- **time\_var** – value
- **cfg** – config file
- **dict\_element** – which section/module the attributes belong to

**Returns** updated config dictionary

`dicom_wsi.mods.utils.uid_maker(k, v, cfg, dict_element='BaseAttributes')`

Make a UID if needed

**Parameters**

- **k** – key
- **v** – value
- **cfg** – config file
- **dict\_element** – which section/module the attributes belong to

**Returns** updated config dictionary





Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 11.1 Types of Contributions

### 11.1.1 Report Bugs

Report bugs at [https://github.com/Steven-N-Hart/dicom\\_wsi/issues](https://github.com/Steven-N-Hart/dicom_wsi/issues).

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 11.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 11.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

## 11.1.4 Write Documentation

dicom-wsi could always use more documentation, whether as part of the official dicom-wsi docs, in docstrings, or even on the web in blog posts, articles, and such.

## 11.1.5 Submit Feedback

The best way to send feedback is to file an issue at [https://github.com/Steven-N-Hart/dicom\\_wsi/issues](https://github.com/Steven-N-Hart/dicom_wsi/issues).

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 11.2 Get Started!

Ready to contribute? Here's how to set up *dicom\_wsi* for local development.

1. Fork the *dicom\_wsi* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dicom_wsi.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv dicom_wsi
$ cd dicom_wsi/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 dicom_wsi tests
$ python setup.py test or pytest
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 11.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.5, 3.6 and 3.7, and for PyPy. Check [https://travis-ci.org/Steven-N-Hart/dicom\\_wsi/pull\\_requests](https://travis-ci.org/Steven-N-Hart/dicom_wsi/pull_requests) and make sure that the tests pass for all supported Python versions.

## 11.4 Tips

To run a subset of tests:

```
$ pytest tests.test_dicom_wsi
```

## 11.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.



### 12.1 Development Lead

- Steven N. Hart, Ph.D. <https://github.com/Steven-N-Hart>

### 12.2 Contributors

- Jin Jiang, Ph.D. <https://github.com/smujiang>



### 13.1 0.1.0 (2019-10-02)

- First release on PyPI.





## CHAPTER 14

---

### Indices and tables

---

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



### d

`dicom_wsi.mods.base_attributes`, [23](#)  
`dicom_wsi.mods.character_validations`,  
    [23](#)  
`dicom_wsi.mods.input_validation`, [23](#)  
`dicom_wsi.mods.pixel_data_conversion`,  
    [24](#)  
`dicom_wsi.mods.pixel_to_slide_conversions`,  
    [24](#)  
`dicom_wsi.mods.sequence_attributes`, [26](#)  
`dicom_wsi.mods.shared_functional_groups`,  
    [26](#)  
`dicom_wsi.mods.utils`, [26](#)



**A**

`add_data()` (in module `dicom_wsi.mods.utils`), 26

`add_imgdata()` (in module `dicom_wsi.mods.pixel_to_slide_conversions`), 24

`add_per_frame_functional_groups_sequence()` (in module `dicom_wsi.mods.pixel_to_slide_conversions`), 24

**B**

`build_base()` (in module `dicom_wsi.mods.base_attributes`), 23

`build_functional_groups()` (in module `dicom_wsi.mods.shared_functional_groups`), 26

`build_sequences()` (in module `dicom_wsi.mods.sequence_attributes`), 26

**C**

`compute_slide_offsets_from_pixel_data()` (in module `dicom_wsi.mods.pixel_to_slide_conversions`), 24

**D**

`define_plane_position_slide_sequence()` (in module `dicom_wsi.mods.pixel_to_slide_conversions`), 25

`dicom_wsi.mods.base_attributes` (module), 23

`dicom_wsi.mods.character_validations` (module), 23

`dicom_wsi.mods.input_validation` (module), 23

`dicom_wsi.mods.pixel_data_conversion` (module), 24

`dicom_wsi.mods.pixel_to_slide_conversions` (module), 24

`dicom_wsi.mods.sequence_attributes` (module), 26

`dicom_wsi.mods.shared_functional_groups` (module), 26

`dicom_wsi.mods.utils` (module), 26

**E**

`ensure_even_image()` (in module `dicom_wsi.mods.pixel_to_slide_conversions`), 25

**G**

`generate_xy_tiles()` (in module `dicom_wsi.mods.pixel_to_slide_conversions`), 25

`get_all_keys()` (in module `dicom_wsi.mods.utils`), 26

`get_info_from_keyword()` (in module `dicom_wsi.mods.utils`), 26

**M**

`make_date()` (in module `dicom_wsi.mods.utils`), 26

`make_datetime()` (in module `dicom_wsi.mods.utils`), 27

`make_time()` (in module `dicom_wsi.mods.utils`), 27

**R**

`resize_wsi_image()` (in module `dicom_wsi.mods.pixel_data_conversion`), 24

**U**

`uid_maker()` (in module `dicom_wsi.mods.utils`), 27

**V**

`validate_cfg()` (in module `dicom_wsi.mods.input_validation`), 23