

# Unittest for media

January 30, 2020

# Contents

<b>1</b>	<b>Test Information</b>	<b>2</b>
1.1	Test Candidate Information . . . . .	2
1.2	Unittest Information . . . . .	2
1.3	Test System Information . . . . .	2
<b>2</b>	<b>Statistic</b>	<b>2</b>
2.1	Test-Statistic for testrun with python 3.6.9 (final) . . . . .	2
2.2	Coverage Statistic . . . . .	3
<b>3</b>	<b>Tested Requirements</b>	<b>4</b>
3.1	Metadata . . . . .	4
3.1.1	Method to get Metadata . . . . .	4
<b>A</b>	<b>Trace for testrun with python 3.6.9 (final)</b>	<b>6</b>
A.1	Tests with status Info (1) . . . . .	6
A.1.1	Method to get Metadata . . . . .	6
<b>B</b>	<b>Test-Coverage</b>	<b>9</b>
B.1	media . . . . .	9
B.1.1	media.__init__.py . . . . .	9
B.1.2	media.metadata.py . . . . .	10

## 1 Test Information

### 1.1 Test Candidate Information

The Module `media` is designed to help on all issues with media files, like tags (e.g. `exif`, `id3`) and transformations. For more Information read the documentation.

---

#### Library Information

Name	media
State	Released
Supported Interpreters	python3
Version	9e6f9c0bef4dee208308971155bc254c

---

#### Dependencies

---

### 1.2 Unittest Information

---

#### Unittest Information

Version	8c004e5e652fd9d0db633e752ce36782
Testruns with	python 3.6.9 (final)

---

### 1.3 Test System Information

---

#### System Information

Architecture	64bit
Distribution	LinuxMint 19.3 tricia
Hostname	ahorn
Kernel	5.3.0-28-generic (#30 18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/media/unittest
System	Linux
Username	dirk

---

## 2 Statistic

### 2.1 Test-Statistic for testrun with python 3.6.9 (final)

---

Number of tests	<b>1</b>
Number of successfull tests	<b>1</b>
Number of possibly failed tests	<b>0</b>
Number of failed tests	<b>0</b>

---

Executionlevel	Full Test (all defined tests)
Time consumption	0.461s

---

## 2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
media	98.6%	97.5%
media.__init__.py	100.0%	
media.metadata.py	98.4%	

## 3 Tested Requirements

### 3.1 Metadata

#### 3.1.1 Method to get Metadata

##### Description

A Method shall return the metadata for a given media filename.

##### Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.1!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/media/unittest/src/tests/_init_.py (20)
Start-Time:	2020-01-30 23:35:34,229
Finished-Time:	2020-01-30 23:35:34,690
Time-Consumption	0.461s
<b>Testsummary:</b>	
<b>Success</b>	Media data for unknown.txt is correct (Content None and Type is <class 'NoneType'>).
<b>Success</b>	Media data for audio.mp3 is correct (Content {'duration': 236.094694, 'bitrate': 290743, 'artist': 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock', 'year': 2016, 'size': 8580366, 'time': 1451606398, 'tm_is_subst': True} and Type is <class 'dict'>).
<b>Success</b>	Media data for audio_fail_conv.mp3 is correct (Content {'duration': 281.991837, 'bitrate': 228298, 'title': 'Video Games (Album Version Remastered)', 'artist': 'Lana Del Rey', 'album': 'Born To Die', 'genre': 'Pop', 'track': 4, 'year': 2012, 'size': 8047290, 'time': 1325375995, 'tm_is_subst': True} and Type is <class 'dict'>).
<b>Success</b>	Media data for image_exif_gps.jpg is correct (Content {'time': 1518783213, 'exposure_program': 'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off', 'aperture': 2.2, 'focal_length': 4.5, 'gps': {'lon': 12.140646934444444, 'lat': 53.68635940527778}, 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968, 'size': 4342955, 'camera': 'HUAWAI: EVA-L09'} and Type is <class 'dict'>).
<b>Success</b>	Media data for image_exif_no_gps.jpg is correct (Content {'time': 1515143529, 'exposure_program': 'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired', 'aperture': 2.2, 'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0, 'width': 2976, 'size': 2837285, 'camera': 'HUAWAI: EVA-L09'} and Type is <class 'dict'>).
<b>Success</b>	Media data for image_non_exif.jpg is correct (Content {'size': 1139092, 'time': 1449870515, 'tm_is_subst': True} and Type is <class 'dict'>).
<b>Success</b>	Media data for image_extraction_failed.jpg is correct (Content {'time': 1226149915, 'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired', 'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1, 'width': 3888, 'size': 1301272, 'camera': 'Canon: Canon EOS 40D'} and Type is <class 'dict'>).
<b>Success</b>	Media data for video.3gp is correct (Content {'width': 800, 'height': 480, 'ratio': 1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size': 1160345} and Type is <class 'dict'>).
<b>Success</b>	Media data for video.mp4 is correct (Content {'width': 1920, 'height': 1080, 'ratio': 1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size': 27838508} and Type is <class 'dict'>).

Unittest for media

**Success** Media data for video\_special\_time.avi is correct (Content {'width': 320, 'height': 240, 'ratio': 0.0, 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size': 2787622} and Type is <class 'dict'>).

**Success** Media data for video\_no\_date.avi is correct (Content {'width': 640, 'height': 480, 'ratio': 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'size': 2965248, 'time': 1158528375, 'tm.is\_subst': True} and Type is <class 'dict'>).

---

## A Trace for testrun with python 3.6.9 (final)

### A.1 Tests with status Info (1)

#### A.1.1 Method to get Metadata

##### Description

A Method shall return the metadata for a given media filename.

##### Testresult

This test was passed with the state: **Success**.

---

**Success** Media data for unknown.txt is correct (Content None and Type is <class 'NoneType'>).

---

Filetype not known: /user\_data/data/dirk/prj/unittest/media/unittest/input\_data/unknown.txt

Result (Media data for unknown.txt): None (<class 'NoneType'>)

Expectation (Media data for unknown.txt): result = None (<class 'NoneType'>)

---

**Success** Media data for audio.mp3 is correct (Content {'duration': 236.094694, 'bitrate': 290743, 'artist': 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock', 'year': 2016, 'size': 8580366, 'time': 1451606398, 'tm\_is\_subst': True} and Type is <class 'dict'>).

---

Result (Media data for audio.mp3): { 'duration': 236.094694, 'bitrate': 290743, 'artist':  
 ↳ 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock', 'year': 2016,  
 ↳ 'size': 8580366, 'time': 1451606398, 'tm\_is\_subst': True } (<class 'dict'>)

Expectation (Media data for audio.mp3): result = { 'duration': 236.094694, 'bitrate': 290743,  
 ↳ 'artist': 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock',  
 ↳ 'year': 2016, 'time': 1451606398, 'tm\_is\_subst': True, 'size': 8580366 } (<class 'dict'>)

---

**Success** Media data for audio\_fail\_conv.mp3 is correct (Content {'duration': 281.991837, 'bitrate': 228298, 'title': 'Video Games (Album Version Remastered)', 'artist': 'Lana Del Rey', 'album': 'Born To Die', 'genre': 'Pop', 'track': 4, 'year': 2012, 'size': 8047290, 'time': 1325375995, 'tm\_is\_subst': True} and Type is <class 'dict'>).

---

Can't convert 'N/A' (bitrate) for bitrate

Result (Media data for audio\_fail\_conv.mp3): { 'duration': 281.991837, 'bitrate': 228298,  
 ↳ 'title': 'Video Games (Album Version Remastered)', 'artist': 'Lana Del Rey', 'album':  
 ↳ 'Born To Die', 'genre': 'Pop', 'track': 4, 'year': 2012, 'size': 8047290, 'time':  
 ↳ 1325375995, 'tm\_is\_subst': True } (<class 'dict'>)

Expectation (Media data for audio\_fail\_conv.mp3): result = { 'duration': 281.991837,  
 ↳ 'bitrate': 228298, 'artist': 'Lana Del Rey', 'title': 'Video Games (Album Version  
 ↳ Remastered)', 'album': 'Born To Die', 'track': 4, 'genre': 'Pop', 'year': 2012, 'time':  
 ↳ 1325375995, 'tm\_is\_subst': True, 'size': 8047290 } (<class 'dict'>)

---

**Success** Media data for image\_exif\_gps.jpg is correct (Content {'time': 1518783213, 'exposure\_program': 'Program Normal', 'exposure\_time': 0.000535, 'flash': 'Auto Off', 'aperture': 2.2, 'focal\_length': 4.5, 'gps': {'lon': 12.140646934444444, 'lat': 53.68635940527778}, 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968, 'size': 4342955, 'camera': 'HUAWEI: EVA-L09'} and Type is <class 'dict'>).

---

```
Result (Media data for image_exif_gps.jpg): { 'time': 1518783213, 'exposure_program':
↳ 'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off', 'aperture': 2.2,
↳ 'focal_length': 4.5, 'gps': { 'lon': 12.140646934444444, 'lat': 53.68635940527778 },
↳ 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968, 'size': 4342955, 'camera':
↳ 'HUAWEI: EVA-L09' } (<class 'dict'>)
```

```
Expectation (Media data for image_exif_gps.jpg): result = { 'time': 1518783213,
↳ 'exposure_program': 'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off',
↳ 'aperture': 2.2, 'focal_length': 4.5, 'gps': { 'lon': 12.140646934444444, 'lat':
↳ 53.68635940527778 }, 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968,
↳ 'camera': 'HUAWEI: EVA-L09', 'size': 4342955 } (<class 'dict'>)
```

---

**Success** Media data for image\_exif\_no\_gps.jpg is correct (Content {'time': 1515143529, 'exposure\_program': 'Program Normal', 'exposure\_time': 0.03, 'flash': 'Fired', 'aperture': 2.2, 'focal\_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0, 'width': 2976, 'size': 2837285, 'camera': 'HUAWEI: EVA-L09'} and Type is <class 'dict'>).

---

```
Result (Media data for image_exif_no_gps.jpg): { 'time': 1515143529, 'exposure_program':
↳ 'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired', 'aperture': 2.2,
↳ 'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0, 'width': 2976, 'size':
↳ 2837285, 'camera': 'HUAWEI: EVA-L09' } (<class 'dict'>)
```

```
Expectation (Media data for image_exif_no_gps.jpg): result = { 'time': 1515143529,
↳ 'exposure_program': 'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired',
↳ 'aperture': 2.2, 'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0,
↳ 'width': 2976, 'camera': 'HUAWEI: EVA-L09', 'size': 2837285 } (<class 'dict'>)
```

---

**Success** Media data for image\_non\_exif.jpg is correct (Content {'size': 1139092, 'time': 1449870515, 'tm\_is\_subst': True} and Type is <class 'dict'>).

---

```
/user_data/data/dirk/prj/unittest/media/unittest/input_data/image_non_exif.jpg does not have
↳ any exif information
```

```
Result (Media data for image_non_exif.jpg): { 'size': 1139092, 'time': 1449870515,
↳ 'tm_is_subst': True } (<class 'dict'>)
```

```
Expectation (Media data for image_non_exif.jpg): result = { 'time': 1449870515,
↳ 'tm_is_subst': True, 'size': 1139092 } (<class 'dict'>)
```

---

**Success** Media data for image\_extraction\_failed.jpg is correct (Content {'time': 1226149915, 'exposure\_program': 'Program Normal', 'exposure\_time': 0.008, 'flash': 'Fill Fired', 'aperture': 7.1, 'focal\_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1, 'width': 3888, 'size': 1301272, 'camera': 'Canon: Canon EOS 40D'} and Type is <class 'dict'>).

---



GPS data extraction failed for {0: b'\x02\x02\x00\x00'}

```
Result (Media data for image_extraction_failed.jpg): { 'time': 1226149915,
↳ 'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired',
↳ 'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1,
↳ 'width': 3888, 'size': 1301272, 'camera': 'Canon: Canon EOS 40D' } (<class 'dict'>)
```

```
Expectation (Media data for image_extraction_failed.jpg): result = { 'time': 1226149915,
↳ 'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired',
↳ 'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1,
↳ 'width': 3888, 'camera': 'Canon: Canon EOS 40D', 'size': 1301272 } (<class 'dict'>)
```

---

**Success** Media data for video.3gp is correct (Content {'width': 800, 'height': 480, 'ratio': 1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size': 1160345} and Type is <class 'dict'>).

```
Result (Media data for video.3gp): { 'width': 800, 'height': 480, 'ratio':
↳ 1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size':
↳ 1160345 } (<class 'dict'>)
```

```
Expectation (Media data for video.3gp): result = { 'width': 800, 'height': 480, 'ratio':
↳ 1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size':
↳ 1160345 } (<class 'dict'>)
```

---

**Success** Media data for video.mp4 is correct (Content {'width': 1920, 'height': 1080, 'ratio': 1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size': 27838508} and Type is <class 'dict'>).

```
Result (Media data for video.mp4): { 'width': 1920, 'height': 1080, 'ratio':
↳ 1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size':
↳ 27838508 } (<class 'dict'>)
```

```
Expectation (Media data for video.mp4): result = { 'width': 1920, 'height': 1080, 'ratio':
↳ 1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size':
↳ 27838508 } (<class 'dict'>)
```

---

**Success** Media data for video\_special\_time.avi is correct (Content {'width': 320, 'height': 240, 'ratio': 0.0, 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size': 2787622} and Type is <class 'dict'>).

Can't convert 'N/A' (duration) for duration

```
Result (Media data for video_special_time.avi): { 'width': 320, 'height': 240, 'ratio': 0.0,
↳ 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size': 2787622 } (<class
↳ 'dict'>)
```

```
Expectation (Media data for video_special_time.avi): result = { 'width': 320, 'height': 240,
↳ 'ratio': 0.0, 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size':
↳ 2787622 } (<class 'dict'>)
```

---

**Success** Media data for video\_no\_date.avi is correct (Content {'width': 640, 'height': 480, 'ratio': 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'size': 2965248, 'time': 1158528375, 'tm\_is\_subst': True} and Type is <class 'dict'>).

```
Result (Media data for video_no_date.avi): { 'width': 640, 'height': 480, 'ratio':
↳ 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'size': 2965248, 'time':
↳ 1158528375, 'tm_is_subst': True } (<class 'dict'>)
```

```
Expectation (Media data for video_no_date.avi): result = { 'width': 640, 'height': 480,
↳ 'ratio': 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'time': 1158528375,
↳ 'tm_is_subst': True, 'size': 2965248 } (<class 'dict'>)
```

## B Test-Coverage

### B.1 media

The line coverage for media was 98.6%

The branch coverage for media was 97.5%

#### B.1.1 media.\_\_init\_\_.py

The line coverage for media.\_\_init\_\_.py was 100.0%

The branch coverage for media.\_\_init\_\_.py was 97.5%

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 media (Media Tools)
6 =====
7
8 **Author:**
9
10 * Dirk Alders <sudo-dirk@mount-mockery.de>
11
12 **Description:**
13
14     This module helps on all issues with media files , like tags (e.g. exif, id3) and
15     transformations.
16
17 **Submodules:**
18
19 * :func:`media.get_media_data`
20
21 **Unittest:**
22
23     See also the :download:`unittest <../../media/_testresults_/unittest.pdf>` documentation.
24 """
25
26 __DEPENDENCIES__ = []
27
28 import logging
29 from media import metadata
30
31 logger_name = 'MEDIA'
32 logger = logging.getLogger(logger_name)
33
34 __DESCRIPTION__ = """The Module {\\tt %s} is designed to help on all issues with media files ,
35     like tags (e.g. exif, id3) and transformations.
36 For more Information read the documentation.""" % __name__.replace('-', '\\-')
37 """The Module Description"""
```

```

36 __INTERPRETER__ = (3. )
37 """The Tested Interpreter - Versions"""
38
39
40 def get_media_data(full_path):
41     ft = metadata.get_filetype(full_path)
42     #
43     if ft == metadata.FILETYPE_AUDIO:
44         return metadata.get_audio_data(full_path)
45     elif ft == metadata.FILETYPE_IMAGE:
46         return metadata.get_image_data(full_path)
47     elif ft == metadata.FILETYPE_VIDEO:
48         return metadata.get_video_data(full_path)
49     else:
50         logger.warning('Filetype not known: %s', full_path)

```

## B.1.2 media.metadata.py

The line coverage for media.metadata.py was 98.4%

The branch coverage for media.metadata.py was 97.5%

```

1 import logging
2 import os
3 from PIL import Image
4 import subprocess
5 import time
6
7
8 logger_name = 'MEDIA'
9 logger = logging.getLogger(logger_name)
10
11 FILETYPE_AUDIO = 'audio'
12 FILETYPE_IMAGE = 'image'
13 FILETYPE_VIDEO = 'video'
14
15 EXTENSIONS_AUDIO = ['.mp3', ]
16 EXTENSIONS_IMAGE = ['.jpg', '.jpeg', '.jpe', '.png', '.tif', '.tiff', '.gif', ]
17 EXTENSIONS_VIDEO = ['.avi', '.mpg', '.mpeg', '.mpe', '.mov', '.qt', '.mp4', '.webm', '.ogv', '.
18     flv', '.3gp', ]
19
20 KEY_ALBUM = 'album'
21 KEY_APERTURE = 'aperture'
22 KEY_ARTIST = 'artist'
23 KEY_BITRATE = 'bitrate'
24 KEY_CAMERA = 'camera'
25 KEY_DURATION = 'duration'
26 KEY_EXPOSURE_PROGRAM = 'exposure_program'
27 KEY_EXPOSURE_TIME = 'exposure_time'
28 KEY_FLASH = 'flash'
29 KEY_FOCAL_LENGTH = 'focal_length'
30 KEY_GENRE = 'genre'
31 KEY_GPS = 'gps'
32 KEY_HEIGHT = 'height'
33 KEY_ISO = 'iso'
34 KEY_ORIENTATION = 'orientation'
35 KEY_RATIO = 'ratio'
36 KEY_SIZE = 'size'
37 KEY_TIME = 'time' # USE time.localtime(value) or datetime.fromtimestamp(value) to convert the
38     timestamp
39 KEY_TIME_IS_SUBSTITUTION = 'tm_is_subst'
40 KEY_TITLE = 'title'
41 KEY_TRACK = 'track'
42 KEY_WIDTH = 'width'
43 KEY_YEAR = 'year'

```

42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102

```

__KEY_CAMERA_VENDOR__ = 'camera_vendor'
__KEY_CAMERA_MODEL__ = 'camera_model'

def get_filetype(full_path):
    ext = os.path.splitext(full_path.lower())[1]
    if ext in EXTENTIONS_AUDIO:
        return FILETYPE_AUDIO
    elif ext in EXTENTIONS_IMAGE:
        return FILETYPE_IMAGE
    elif ext in EXTENTIONS_VIDEO:
        return FILETYPE_VIDEO

def get_audio_data(full_path):
    conv_key_dict = {}
    conv_key_dict['album'] = (str, KEY_ALBUM)
    conv_key_dict['TAG:album'] = (str, KEY_ALBUM)
    conv_key_dict['TAG:artist'] = (str, KEY_ARTIST)
    conv_key_dict['artist'] = (str, KEY_ARTIST)
    conv_key_dict['bit_rate'] = (int_conv, KEY_BITRATE)
    conv_key_dict['duration'] = (float, KEY_DURATION)
    conv_key_dict['TAG:genre'] = (str, KEY_GENRE)
    conv_key_dict['genre'] = (str, KEY_GENRE)
    conv_key_dict['TAG:title'] = (str, KEY_TITLE)
    conv_key_dict['title'] = (str, KEY_TITLE)
    conv_key_dict['TAG:track'] = (int_conv, KEY_TRACK)
    conv_key_dict['track'] = (int_conv, KEY_TRACK)
    conv_key_dict['TAG:date'] = (int_conv, KEY_YEAR)
    conv_key_dict['date'] = (int_conv, KEY_YEAR)
    return __adapt_data__(__get_xprobe_data__(full_path, conv_key_dict), full_path)

def get_video_data(full_path):
    conv_key_dict = {}
    conv_key_dict['creation_time'] = (vid_datetime_conv, KEY_TIME)
    conv_key_dict['TAG:creation_time'] = (vid_datetime_conv, KEY_TIME)
    conv_key_dict['bit_rate'] = (int_conv, KEY_BITRATE)
    conv_key_dict['duration'] = (float, KEY_DURATION)
    conv_key_dict['height'] = (int_conv, KEY_HEIGHT)
    conv_key_dict['width'] = (int_conv, KEY_WIDTH)
    conv_key_dict['display_aspect_ratio'] = (ratio_conv, KEY_RATIO)
    return __adapt_data__(__get_xprobe_data__(full_path, conv_key_dict), full_path)

def get_image_data(full_path):
    return __adapt_data__(__get_exif_data__(full_path), full_path)

def __adapt_data__(data, full_path):
    data[KEY_SIZE] = os.path.getsize(full_path)
    # Join Camera Vendor and Camera Model
    if __KEY_CAMERA_MODEL__ in data and __KEY_CAMERA_VENDOR__ in data:
        model = data.pop(__KEY_CAMERA_MODEL__)
        vendor = data.pop(__KEY_CAMERA_VENDOR__)
        data[KEY_CAMERA] = '%s: %s' % (vendor, model)
    # Add time if not exists
    if KEY_TIME not in data:
        if KEY_YEAR in data and KEY_TRACK in data:
            # Use a date where track 1 is the newest in the given year

```

## Unittest for media

```

103     minute = int(data[KEY_TRACK] / 60)
104     second = (data[KEY_TRACK] - 60 * minute) % 60
105     #
106     data[KEY_TIME] = int(time.mktime((data[KEY_YEAR], 1, 1, 0, 59 - minute, 59 - second,
107     0, 0, 0)))
108     data[KEY_TIME_IS_SUBSTITUTION] = True
109     else:
110     data[KEY_TIME] = int(os.path.getmtime(full_path))
111     data[KEY_TIME_IS_SUBSTITUTION] = True
112     return data
113
114 def __get_xxprobe_data__(full_path, conv_key_dict):
115     def _ffprobe_command(full_path):
116         return ['ffprobe', '-v', 'quiet', '-show_format', '-show_streams', full_path]
117
118     def _avprobe_command(full_path):
119         return ['avprobe', '-v', 'quiet', '-show_format', '-show_streams', full_path]
120
121     try:
122         xxprobe_text = subprocess.check_output(_avprobe_command(full_path))
123     except FileNotFoundError:
124         try:
125             xxprobe_text = subprocess.check_output(_ffprobe_command(full_path))
126         except FileNotFoundError:
127             logger.warning('ffprobe and avprobe seem to be not installed')
128         return {}
129     #
130     rv = {}
131     for line in xxprobe_text.decode('utf-8').splitlines():
132         try:
133             key, val = [snippet.strip() for snippet in line.split('=')]
134         except ValueError:
135             continue
136         else:
137             if key in conv_key_dict:
138                 tp, name = conv_key_dict[key]
139                 try:
140                     rv[name] = tp(val)
141                 except ValueError:
142                     logger.log(logging.WARNING if val else logger.INFO, 'Can\'t convert %s (%s)
143     for %s', repr(val), name, name)
144     return rv
145
146 def __get_exif_data__(full_path):
147     rv = {}
148     im = Image.open(full_path)
149     try:
150         exif = dict(im._getexif().items())
151     except AttributeError:
152         logger.debug('%s does not have any exif information', full_path)
153     else:
154         conv_key_dict = {}
155         # IMAGE
156         conv_key_dict[0x9003] = (__datetime_conv__, KEY_TIME)
157         conv_key_dict[0x8822] = (__exposure_program_conv__, KEY_EXPOSURE_PROGRAM)
158         conv_key_dict[0x829A] = (__num_denum_conv__, KEY_EXPOSURE_TIME)
159         conv_key_dict[0x9209] = (__flash_conv__, KEY_FLASH)
160         conv_key_dict[0x829D] = (__num_denum_conv__, KEY_APERTURE)
161         conv_key_dict[0x920A] = (__num_denum_conv__, KEY_FOCAL_LENGTH)
162         conv_key_dict[0x8825] = (__gps_conv__, KEY_GPS)

```

```

163     conv_key_dict[0xA003] = (__int_conv__, KEY_HEIGHT)
164     conv_key_dict[0x8827] = (__int_conv__, KEY_ISO)
165     conv_key_dict[0x010F] = (str, __KEY_CAMERA_VENDOR__)
166     conv_key_dict[0x0110] = (str, __KEY_CAMERA_MODEL__)
167     conv_key_dict[0x0112] = (__int_conv__, KEY_ORIENTATION)
168     conv_key_dict[0xA002] = (__int_conv__, KEY_WIDTH)
169     for key in conv_key_dict:
170         if key in exif:
171             tp, name = conv_key_dict[key]
172             value = tp(exif[key])
173             if value is not None:
174                 rv[name] = value
175     return rv

```

```

176
177
178 # TODO: Join datetime converter __datetime_conv__ and __vid_datetime_conv__

```

```

179 def __datetime_conv__(dt):
180     format_string = "%Y:%m:%d %H:%M:%S"
181     return int(time.mktime(time.strptime(dt, format_string)))

```

```

182
183
184 def __vid_datetime_conv__(dt):
185     try:
186         dt = dt[:dt.index('.')]
187     except ValueError:
188         pass # time string seems to have no '.'
189     dt = dt.replace('T', ' ').replace('/', ' ').replace('\\', ' ')
190     if len(dt) == 16:
191         dt += ':00'
192     format_string = "%Y-%m-%d %H:%M:%S"
193     return int(time.mktime(time.strptime(dt, format_string)))

```

```

194
195
196 def __exposure_program_conv__(n):
197     return {
198         0: 'Unidentified',
199         1: 'Manual',
200         2: 'Program Normal',
201         3: 'Aperture Priority',
202         4: 'Shutter Priority',
203         5: 'Program Creative',
204         6: 'Program Action',
205         7: 'Portrait Mode',
206         8: 'Landscape Mode'
207     }.get(n, None)

```

```

208
209
210 def __flash_conv__(n):
211     return {
212         0: 'No',
213         1: 'Fired',
214         5: 'Fired (?)', # no return sensed
215         7: 'Fired (!)', # return sensed
216         9: 'Fill Fired',
217         13: 'Fill Fired (?)',
218         15: 'Fill Fired (!)',
219         16: 'Off',
220         24: 'Auto Off',
221         25: 'Auto Fired',
222         29: 'Auto Fired (?)',
223         31: 'Auto Fired (!)',

```

```

224         32: 'Not Available'
225     }.get(n, None)
226
227
228 def __int_conv__(value):
229     try:
230         return int(value)
231     except ValueError:
232         for c in ['.', '/', '-']:
233             p = value.find(c)
234             if p >= 0:
235                 value = value[:p]
236         return int(value)
237
238
239 def __num_denum_conv__(data):
240     num, denum = data
241     return num / denum
242
243
244 def __gps_conv__(data):
245     def lat_lon_cal(lon_or_lat):
246         lon_lat = 0.
247         fac = 1.
248         for num, denum in lon_or_lat:
249             lon_lat += float(num) / float(denum) * fac
250             fac *= 1. / 60.
251         return lon_lat
252     try:
253         lon = lat_lon_cal(data[0x0004])
254         lat = lat_lon_cal(data[0x0002])
255         if lon != 0 or lat != 0: # do not use lon and lat equal 0, caused by motorola gps
256             return {'lon': lon, 'lat': lat}
257     except KeyError:
258         logger.warning('GPS data extraction failed for %s', repr(data))
259
260
261 def __ratio_conv__(ratio):
262     ratio = ratio.replace('\\', '/')
263     num, denum = ratio.split(':')
264     return float(num) / float(denum)

```