
leicacam Documentation

Release 0.4.2

Arve Seljebu

Jun 10, 2020

CONTENTS

1	leicacam package	3
1.1	Submodules	5
1.2	leicacam.async_cam module	5
1.3	leicacam.cam module	6
	Python Module Index	9
	Index	11

- [Project source at GitHub](#)
- [Releases at PyPI](#)

Contents:

LEICACAM PACKAGE

Control Leica microscopes with python.

```
class leicacam.CAM(*args, **kwargs)
```

Bases: *leicacam.cam.BaseCAM*

Driver for LASAF Computer Assisted Microscopy.

```
autofocus_scan()
```

Start the autofocus job.

```
close()
```

Close the socket.

```
connect()
```

Connect to LASAF through a CAM-socket.

```
disable(slide=0, wellx=1, welly=1, fieldx=1, fieldy=1)
```

Disable a given scan field.

```
disable_all()
```

Disable all scan fields.

```
enable(slide=0, wellx=1, welly=1, fieldx=1, fieldy=1)
```

Enable a given scan field.

```
enable_all()
```

Enable all scan fields.

```
flush()
```

Flush incoming socket messages.

```
get_information(about='stage')
```

Get information about given keyword. Defaults to stage.

```
load_template(filename='{ScanningTemplate}leicacam.xml')
```

Load scanning template from filename.

Template needs to exist in database, otherwise it will not load.

Parameters filename (*str*) – Filename to template to load. Filename may contain path also, in such case, the basename will be used. '.xml' will be stripped from the filename if it exists because of a bug; LASAF implicit add '.xml'. If '{ScanningTemplate}' is omitted, it will be added.

Returns Response from LASAF in an ordered dict.

Return type collections.OrderedDict

Example

```
>>> # load {ScanningTemplate}leicacam.xml
>>> cam.load_template('leicacam')

>>> # load {ScanningTemplate}leicacam.xml
>>> cam.load_template('{ScanningTemplate}leicacam')

>>> # load {ScanningTemplate}leicacam.xml
>>> cam.load_template('/path/to/{ScanningTemplate}leicacam.xml')
```

pause_scan()

Pause the matrix scan.

receive()

Receive message from socket interface as list of OrderedDict.

save_template(filename='{ScanningTemplate}leicacam.xml')

Save scanning template to filename.

send(commands)

Send commands to LASAF through CAM-socket.

Parameters **commands** (*list of tuples or bytes string*) – Commands as a list of tuples or a bytes string. cam.prefix is always prepended before sending.

Returns Bytes sent.

Return type int

Example

```
>>> # send list of tuples
>>> cam.send([('cmd', 'enableall'), ('value', 'true')])

>>> # send bytes string
>>> cam.send(b'/cmd:enableall /value:true')
```

start_scan()

Start the matrix scan.

stop_scan()

Stop the matrix scan.

wait_for(cmd, value=None, timeout=60)

Hang until command is received.

If value is supplied, it will hang until cmd:value is received.

Parameters

- **cmd** (*string*) – Command to wait for in bytestring from microscope CAM interface. If value is falsy, value of received command does not matter.
- **value** (*string*) – Wait until cmd:value is received.
- **timeout** (*int*) – Minutes to wait for command. If timeout is reached, an empty OrderedDict will be returned.

Returns Last received message or empty message if timeout is reached.

Return type collections.OrderedDict

1.1 Submodules

1.2 leicacam.async_cam module

Provide an interface using asyncio to the CAM server.

class leicacam.async_cam.**AsyncCAM** (*args, loop=None, **kwargs)

Bases: *leicacam.cam.BaseCAM*

Driver for LASAF Computer Assisted Microscopy using asyncio.

close()

Close stream.

async connect()

Connect to LASAF through a CAM-socket.

async receive()

Receive message from socket interface as list of OrderedDict.

async send(commands)

Send commands to LASAF through CAM-socket.

Parameters **commands** (*list of tuples or bytes string*) – Commands as a list of tuples or a bytes string. cam.prefix is allways prepended before sending.

Returns Bytes sent.

Return type int

Example

```
>>> # send list of tuples
>>> await cam.send([('cmd', 'enableall'), ('value', 'true')])

>>> # send bytes string
>>> await cam.send(b'/cmd:enableall /value:true')
```

async wait_for(cmd, value=None, timeout=60)

Hang until command is received.

If value is supplied, it will hang until cmd:value is received.

Parameters

- **cmd** (*string*) – Command to wait for in bytestring from microscope CAM interface. If value is falsy, value of received command does not matter.
- **value** (*string*) – Wait until cmd:value is received.
- **timeout** (*int*) – Minutes to wait for command. If timeout is reached, an empty OrderedDict will be returned.

Returns Last received message or empty message if timeout is reached.

Return type collections.OrderedDict

1.3 leicacam.cam module

Provide an interface to the CAM server.

```
class leicacam.cam.BaseCAM (host='127.0.0.1', port=8895)  
    Bases: object
```

Base driver for LASAF Computer Assisted Microscopy.

```
class leicacam.cam.CAM (*args, **kwargs)  
    Bases: leicacam.cam.BaseCAM
```

Driver for LASAF Computer Assisted Microscopy.

```
autofocus_scan ()  
    Start the autofocus job.
```

```
close ()  
    Close the socket.
```

```
connect ()  
    Connect to LASAF through a CAM-socket.
```

```
disable (slide=0, wellx=1, welly=1, fieldx=1, fieldy=1)  
    Disable a given scan field.
```

```
disable_all ()  
    Disable all scan fields.
```

```
enable (slide=0, wellx=1, welly=1, fieldx=1, fieldy=1)  
    Enable a given scan field.
```

```
enable_all ()  
    Enable all scan fields.
```

```
flush ()  
    Flush incoming socket messages.
```

```
get_information (about='stage')  
    Get information about given keyword. Defaults to stage.
```

```
load_template (filename='{ScanningTemplate}leicacam.xml')  
    Load scanning template from filename.
```

Template needs to exist in database, otherwise it will not load.

Parameters **filename** (*str*) – Filename to template to load. Filename may contain path also, in such case, the basename will be used. ‘.xml’ will be stripped from the filename if it exists because of a bug; LASAF implicit add ‘.xml’. If ‘{ScanningTemplate}’ is omitted, it will be added.

Returns Response from LASAF in an ordered dict.

Return type collections.OrderedDict

Example

```
>>> # load {ScanningTemplate}leicacam.xml
>>> cam.load_template('leicacam')

>>> # load {ScanningTemplate}leicacam.xml
>>> cam.load_template('{ScanningTemplate}leicacam')

>>> # load {ScanningTemplate}leicacam.xml
>>> cam.load_template('/path/to/{ScanningTemplate}leicacam.xml')
```

pause_scan()

Pause the matrix scan.

receive()

Receive message from socket interface as list of OrderedDict.

save_template(filename='{ScanningTemplate}leicacam.xml')

Save scanning template to filename.

send(commands)

Send commands to LASAF through CAM-socket.

Parameters **commands** (*list of tuples or bytes string*) – Commands as a list of tuples or a bytes string. cam.prefix is always prepended before sending.

Returns Bytes sent.

Return type int

Example

```
>>> # send list of tuples
>>> cam.send([('cmd', 'enableall'), ('value', 'true')])

>>> # send bytes string
>>> cam.send(b'/cmd:enableall /value:true')
```

start_scan()

Start the matrix scan.

stop_scan()

Stop the matrix scan.

wait_for(cmd, value=None, timeout=60)

Hang until command is received.

If value is supplied, it will hang until cmd:value is received.

Parameters

- **cmd** (*string*) – Command to wait for in bytestring from microscope CAM interface. If value is falsy, value of received command does not matter.
- **value** (*string*) – Wait until cmd:value is received.
- **timeout** (*int*) – Minutes to wait for command. If timeout is reached, an empty OrderedDict will be returned.

Returns Last received message or empty message if timeout is reached.

Return type collections.OrderedDict

`leicacam.cam.bytes_as_dict(msg)`

Parse CAM message to OrderedDict based on format /key:val.

Parameters `msg` (*bytes*) – Sequence of /key:val.

Returns With /key:val => dict[key] = val.

Return type collections.OrderedDict

`leicacam.cam.check_messages(msgs, cmd, value=None)`

Check if specific message is present.

Parameters

- **cmd** (*string*) – Command to check for in bytestring from microscope CAM interface. If value is falsy, value of received command does not matter.
- **value** (*string*) – Check if cmd:value is received.

Returns Correct message or None if no correct message is found.

Return type collections.OrderedDict

`leicacam.cam.logger(function)`

Decorate passed in function and log message to module logger.

`leicacam.cam.tuples_as_bytes(cmds)`

Format list of tuples to CAM message with format /key:val.

Parameters `cmds` (*list of tuples*) – List of commands as tuples.

Returns Sequence of /key:val.

Return type bytes

Example

```
>>> tuples_as_bytes([('cmd', 'val'), ('cmd2', 'val2')])
b'/cmd:val /cmd2:val2'
```

`leicacam.cam.tuples_as_dict(_list)`

Translate a list of tuples to OrderedDict with key and val as strings.

Parameters `_list` (*list of tuples*) –

Returns

Return type collections.OrderedDict

Example

```
>>> tuples_as_dict([('cmd', 'val'), ('cmd2', 'val2')])
OrderedDict([('cmd', 'val'), ('cmd2', 'val2')])
```

PYTHON MODULE INDEX

I

`leicacam`, 3
`leicacam.async_cam`, 5
`leicacam.cam`, 6

A

AsyncCAM (class in leicacam.async_cam), 5
 autofocus_scan() (leicacam.CAM method), 3
 autofocus_scan() (leicacam.cam.CAM method), 6

B

BaseCAM (class in leicacam.cam), 6
 bytes_as_dict() (in module leicacam.cam), 8

C

CAM (class in leicacam), 3
 CAM (class in leicacam.cam), 6
 check_messages() (in module leicacam.cam), 8
 close() (leicacam.async_cam.AsyncCAM method), 5
 close() (leicacam.CAM method), 3
 close() (leicacam.cam.CAM method), 6
 connect() (leicacam.async_cam.AsyncCAM method), 5
 connect() (leicacam.CAM method), 3
 connect() (leicacam.cam.CAM method), 6

D

disable() (leicacam.CAM method), 3
 disable() (leicacam.cam.CAM method), 6
 disable_all() (leicacam.CAM method), 3
 disable_all() (leicacam.cam.CAM method), 6

E

enable() (leicacam.CAM method), 3
 enable() (leicacam.cam.CAM method), 6
 enable_all() (leicacam.CAM method), 3
 enable_all() (leicacam.cam.CAM method), 6

F

flush() (leicacam.CAM method), 3
 flush() (leicacam.cam.CAM method), 6

G

get_information() (leicacam.CAM method), 3
 get_information() (leicacam.cam.CAM method), 6

L

leicacam
 module, 3
 leicacam.async_cam
 module, 5
 leicacam.cam
 module, 6
 load_template() (leicacam.CAM method), 3
 load_template() (leicacam.cam.CAM method), 6
 logger() (in module leicacam.cam), 8

M

module
 leicacam, 3
 leicacam.async_cam, 5
 leicacam.cam, 6

P

pause_scan() (leicacam.CAM method), 4
 pause_scan() (leicacam.cam.CAM method), 7

R

receive() (leicacam.async_cam.AsyncCAM method), 5
 receive() (leicacam.CAM method), 4
 receive() (leicacam.cam.CAM method), 7

S

save_template() (leicacam.CAM method), 4
 save_template() (leicacam.cam.CAM method), 7
 send() (leicacam.async_cam.AsyncCAM method), 5
 send() (leicacam.CAM method), 4
 send() (leicacam.cam.CAM method), 7
 start_scan() (leicacam.CAM method), 4
 start_scan() (leicacam.cam.CAM method), 7
 stop_scan() (leicacam.CAM method), 4
 stop_scan() (leicacam.cam.CAM method), 7

T

tuples_as_bytes() (in module leicacam.cam), 8
 tuples_as_dict() (in module leicacam.cam), 8

W

`wait_for()` (*leicacam.async_cam.AsyncCAM method*), [5](#)
`wait_for()` (*leicacam.CAM method*), [4](#)
`wait_for()` (*leicacam.cam.CAM method*), [7](#)