pytemscript

Release 3.0a3

Tore Niermann, Grigory Sharov

Feb 08, 2023

CONTENTS

The `pytemscript` package provides a Python wrapper for both standard and advanced scripting interfaces of Thermo Fisher Scientific and FEI microscopes. The functionality is limited to the functionality of the original scripting interfaces. For detailed information about TEM scripting see the documentation accompanying your microscope.

Within the `pytemscript` package two implementations for the high level microscope interface are provided: one for running scripts directly on the microscope PC and one to run scripts remotely over network (not yet available).

Currently the `pytemscript` package requires Python 3.4 or higher. The current plan is to keep the minimum supported Python version at 3.4, since this is the latest Python version supporting Windows XP.

This is a GPL fork of the original BSD-licensed project: https://github.com/niermann/temscript New changes and this whole product is distributed under either version 3 of the GPL License, or (at your option) any later version.

CHAPTER

ONE

DOCUMENTATION

The documentation can be found at https://pytemscript.readthedocs.io

1.1 About

1.1.1 The COM interface

The methods and classes represent the COM objects exposed by the *Scripting* interface. The interface is described in detail in the scripting manual of your microscope (usually in the file `scripting.pdf` located in the `C:\Titan\Tem_help\manual` or `C:\Tecnai\tem_help\manual` directories). Advanced scripting manual can be found in `C:\Titan\Scripting\Advanced TEM Scripting User Guide.pdf`.

The manual is your ultimate reference, this documentation will only describe the python wrapper to the COM interface.

1.1.2 Microscope class

The *Microscope class* class provides the main interface to the microscope.

1.1.3 Enumerations

Many of the attributes return values from enumerations. The complete list can be found in the *Enumerations* section.

1.1.4 Vectors

Some object attributes handle two dimensional vectors (e.g. `ImageShift`). These attributes return `(x, y)` tuples and expect iterable objects (`tuple`, `list`, ...) with two floats when written (numpy arrays with two entries also work).

```
beam_pos = microscope.optics.illumination.beam_shift
print(beam_pos)
(0.0, 0.0)
new_beam_pos = beam_pos[0], beam_pos[1] + 1.02
microscope.optics.illumination.beam_shift = new_beam_pos
```

1.2 Microscope class

The `Microscope` class provides a Python interface to the microscope. Below are the main class properties, each represented by a separate class:

- acquisition = *Acquisition()*
- apertures = *Apertures()*
- autoloader = *Autoloader()*
- detectors = *Detectors()*
- energy_filter = *EnergyFilter()*
- gun = *Gun()*
- optics = *Optics()*
  - illumination = *Illumination()*
  - projection = *Projection()*
- piezo_stage = *PiezoStage()*
- stage = *Stage()*
- stem = *Stem()*
- temperature = *Temperature()*
- user_door = *UserDoor()*
- vacuum = *Vacuum()*

1.2.1 Image object

Two acquisition functions: *acquire_tem_image()* and *acquire_stem_image()* return an `Image` object that has the following methods and properties:

class pytemscript.microscope.**Image**(*obj*, *name=None*, *isAdvanced=False*, ***kwargs*)

Acquired image object.

property bit_depth

Bit depth.

property data

Returns actual image object as numpy int32 array.

property height

Image height in pixels.

property metadata

Returns a metadata dict for advanced camera image.

property name

Image name.

property pixel_type

Image pixels type: uint, int or float.

save(*filename*, *normalize=False*)

Save acquired image to a file.

Parameters

- **filename** (*str*) – File path
- **normalize** (*bool*) – Normalize image, only for non-MRC format

property width

Image width in pixels.

1.2.2 Example usage

```
microscope = Microscope()
curr_pos = microscope.stage.position
print(curr_pos['Y'])
24.05
microscope.stage.move_to(x=-30, y=25.5)

beam_shift = microscope.optics.illumination.beam_shift
defocus = microscope.optics.projection.defocus
microscope.optics.normalize_all()
```

1.2.3 Documentation

class pytemscript.microscope.**Acquisition**(*microscope*)

Image acquisition functions.

In order for acquisition to be available TIA (TEM Imaging and Acquisition) must be running (even if you are using DigitalMicrograph as the CCD server).

If it is necessary to update the acquisition object (e.g. when the STEM detector selection on the TEM UI has been changed), you have to release and recreate the main microscope object. If you do not do so, you keep accessing the same acquisition object which will not work properly anymore.

acquire_film(*film_text*, *exp_time*, ***kwargs*)

Expose a film.

Parameters

- **film_text** (*str*) – Film text, 96 symbols
- **exp_time** (*float*) – Exposure time in seconds

acquire_stem_image(*cameraName*, *size*, *dwell_time=1e-05*, *binning=1*, ***kwargs*)

Acquire a STEM image.

Parameters

- **cameraName** (*str*) – Camera name
- **size** (*IntEnum*) – Image size (AcqImageSize enum)
- **dwell_time** (*float*) – Dwell time in seconds. The frame time equals the dwell time times the number of pixels plus some overhead (typically 20%, used for the line flyback)
- **binning** (*int*) – Binning factor

- **brightness** (*float*) – Brightness setting
- **contrast** (*float*) – Contrast setting

Returns
Image object

acquire_tem_image(*cameraName*, *size=AcqImageSize.FULL*, *exp_time=1*, *binning=1*, ***kwargs*)

Acquire a TEM image.

Parameters

- **cameraName** (*str*) – Camera name
- **size** (*IntEnum*) – Image size (AcqImageSize enum)
- **exp_time** (*float*) – Exposure time in seconds
- **binning** – Binning factor
- **align_image** (*bool*) – Whether frame alignment (i.e. drift correction) is to be applied to the final image as well as the intermediate images. Advanced cameras only.
- **electron_counting** (*bool*) – Use counting mode. Advanced cameras only.
- **eer** (*bool*) – Use EER mode. Advanced cameras only.
- **frame_ranges** (*list*) – List of tuple frame ranges that define the intermediate images, e.g. [(1,2), (2,3)]. Advanced cameras only.
- **use_tecnaiccd** (*bool*) – Use Tecnai CCD plugin to acquire image via Digital Micrograph, only for Gatan cameras. Requires Microscope() initialized with useTecnaiCCD=True

Returns
Image object

Usage:

```
>>> microscope = Microscope()
>>> acq = microscope.acquisition
>>> img = acq.acquire_tem_image("BM-Falcon", AcqImageSize.FULL, exp_time=5.
↪0, binning=1, electron_counting=True, align_image=True)
>>> img.save("aligned_sum.mrc")
>>> print(img.width)
4096
```

class pytemscript.microscope.**Apertures**(*microscope*)

Apertures and VPP controls.

select(*aperture*, *size*)

Select a specific aperture.

Parameters

- **aperture** (*str*) – Aperture name (C1, C2, C3, OBJ or SA)
- **size** (*float*) – Aperture size

property show_all

Returns a dict with apertures information.

vpp_next_position()

Goes to the next preset location on the VPP aperture.

property vpp_position

Returns the index of the current VPP preset position.

class pytemscript.microscope.**Autoloader**(*microscope*)

Sample loading functions.

buffer_cycle()

Synchronously runs the Autoloader buffer cycle.

dock_cassette()

Moves the cassette from the capsule to the docker.

initialize()

Initializes / Recovers the Autoloader for further use.

property is_available

Status of the autoloader. Should be always False on Tecnai instruments.

load_cartridge(*slot*)

Loads the cartridge in the given slot into the microscope.

Parameters
slot (*int*) – Slot number

property number_of_slots

The number of slots in a cassette.

run_inventory()

Performs an inventory of the cassette. Note: This function takes considerable time to execute.

slot_status(*slot*)

The status of the slot specified.

Parameters
slot (*int*) – Slot number

undock_cassette()

Moves the cassette from the docker to the capsule.

unload_cartridge()

Unloads the cartridge currently in the microscope and puts it back into its slot in the cassette.

class pytemscript.microscope.**Detectors**(*microscope*)

CCD/DDD, film/plate and STEM detectors.

property cameras

Returns a dict with parameters for all cameras.

property film_settings

Returns a dict with film settings. Note: The plate camera has become obsolete with Win7 so most of the existing functions are no longer supported.

property screen

Fluorescent screen position. (read/write)

property stem_detectors

Returns a dict with STEM detectors parameters.

class pytemscript.microscope.**EnergyFilter**(*microscope*)

Energy filter controls.

property ht_shift

Returns High Tension energy shift in eV.

insert_slit(*width*)

Insert energy slit.

Parameters
width (*float*) – Slit width in eV

retract_slit()

Retract energy slit.

property slit_width

Returns energy slit width in eV.

property zlp_shift

Returns Zero-Loss Peak (ZLP) energy shift in eV.

class pytemscript.microscope.**Gun**(*microscope*)

Gun functions.

property beam_current

Returns the C-FEG beam current in Amperes.

do_flashing(*flash_type*)

Perform cold FEG flashing.

Parameters
flash_type (*IntEnum*) – FEG flashing type (FegFlashingType enum)

property extractor_voltage

Returns the extractor voltage.

property feg_state

FEG emitter status.

property focus_index

Returns coarse and fine gun lens index.

property ht_state

High tension state: on, off or disabled. Disabling/enabling can only be done via the button on the system on/off-panel, not via script. When switching on the high tension, this function cannot check if and when the set value is actually reached. (read/write)

property shift

Gun shift. (read/write)

property tilt

Gun tilt. (read/write)

property voltage

The value of the HT setting as displayed in the TEM user interface. Units: kVolts. (read/write)

property voltage_max

The maximum possible value of the HT on this microscope. Units: kVolts.

property voltage_offset

High voltage offset. (read/write)

property voltage_offset_range

Returns the high voltage offset range.

class pytemscript.microscope.**Illumination**(*tem*)

Illumination functions.

property C3ImageDistanceParallelOffset

C3 image distance parallel offset. Works only on 3-condenser lens systems. (read/write)

property beam_shift

Beam shift X and Y in um. (read/write)

property beam_tilt

Dark field beam tilt relative to the origin stored at alignment time. Only operational if dark field mode is active. Units: mrad, either in Cartesian (x,y) or polar (conical) tilt angles. The accuracy of the beam tilt physical units depends on a calibration of the tilt angles. (read/write)

property condenser_mode

Mode of the illumination system: parallel or probe. (read/write)

property condenser_stigmator

C2 condenser stigmator X and Y. (read/write)

property convergence_angle

Convergence angle. Works only on 3-condenser lens systems. (read/write)

property dark_field

Dark field mode: cartesian, conical or off. (read/write)

property illuminated_area

Illuminated area. Works only on 3-condenser lens systems. (read/write)

property intensity

Intensity / C2 condenser lens value. (read/write)

property intensity_limit

Intensity limit. Set to False to disable. (read/write)

property intensity_zoom

Intensity zoom. Set to False to disable. (read/write)

property mode

Illumination mode: microprobe or nanoprobe. (read/write)

property probe_defocus

Probe defocus. Works only on 3-condenser lens systems. (read/write)

property rotation_center

Rotation center X and Y in mrad. (read/write) Depending on the scripting version, the values might need scaling by 6.0 to get mrads.

property spotsize

Spotsize number, usually 1 to 11. (read/write)

class pytemscript.microscope.LowDose(*microscope*)

Low Dose functions.

property is_active

Check if the Low Dose is ON.

property is_available

Return True if Low Dose is available.

off()

Switch OFF Low Dose.

on()

Switch ON Low Dose.

property state

Low Dose state (LDState enum). (read/write)

class pytemscript.microscope.Microscope(*useLD=True*, *useTecnaiCCD=False*, *useSEMCCD=False*, *remote=False*)

High level interface to the local microscope. Creating an instance of this class already queries COM interfaces for the instrument.

Parameters

- **useLD** (*bool*) – Connect to LowDose server on microscope PC (limited control only)
- **useTecnaiCCD** (*bool*) – Connect to TecnaiCCD plugin on microscope PC that controls Digital Micrograph (may be faster than via TIA / std scripting)
- **useSEMCCD** (*bool*) – Connect to SerialEMCCD plugin on Gatan PC that controls Digital Micrograph (may be faster than via TIA / std scripting)

property condenser_system

Returns the type of condenser lens system: two or three lenses.

property family

Returns the microscope product family / platform.

property user_buttons

Returns a dict with assigned hand panels buttons.

class pytemscript.microscope.Optics(*microscope*)

Projection, Illumination functions.

beam_blank()

Activates the beam blanker.

beam_unblank()

Deactivates the beam blanker.

property is_autonormalize_on

Status of the automatic normalization procedures performed by the TEM microscope. Normally they are active, but for scripting it can be convenient to disable them temporarily.

property is_beam_blanked

Status of the beam blanker.

property is_shutter_override_on

Determines the state of the shutter override function. WARNING: Do not leave the Shutter override on when stopping the script. The microscope operator will be unable to have a beam come down and has no separate way of seeing that it is blocked by the closed microscope shutter.

normalize(*mode*)

Normalize condenser or projection lens system. :param mode: Normalization mode (ProjectionNormalization or IlluminationNormalization enum) :type mode: IntEnum

normalize_all()

Normalize all lenses.

property screen_current

The current measured on the fluorescent screen (units: nanoAmperes).

class pytemscript.microscope.**PiezoStage**(*microscope*)

Piezo stage functions.

property position

The current position of the piezo stage (x,y,z in um).

property position_range

Return min and max positions.

property velocity

Returns a dict with stage velocities.

class pytemscript.microscope.**Projection**(*projection*)

Projection system functions.

property camera_length

The reference camera length in m (screen up setting).

property defocus

Defocus value in um. (read/write)

property detector_shift

Detector shift. (read/write)

property detector_shift_mode

Detector shift mode. (read/write)

property diffraction_shift

Diffraction shift in mrad. (read/write)

property diffraction_stigmator

Diffraction stigmator. (read/write)

eftem_off()

Switch off EFTEM.

eftem_on()

Switch on EFTEM.

property focus

Absolute focus value. (read/write)

property image_beam_shift

Image shift with beam shift compensation in um. (read/write)

property image_beam_tilt

Beam tilt with diffraction shift compensation in mrad. (read/write)

property image_rotation

The rotation of the image or diffraction pattern on the fluorescent screen with respect to the specimen. Units: mrad.

property image_shift

Image shift in um. (read/write)

property is_eftem_on

Check if the EFTEM lens program setting is ON.

property magnification

The reference magnification value (screen up setting).

property magnification_range

Submode of the projection system (either LM, M, SA, MH, LAD or D). The imaging submode can change when the magnification is changed.

property mode

Main mode of the projection system (either imaging or diffraction). (read/write)

property objective_stigmator

Objective stigmator. (read/write)

reset_defocus()

Reset defocus value in the TEM user interface to zero. Does not change any lenses.

class pytemscript.microscope.**Stage**(*microscope*)

Stage functions.

go_to(***kwargs*)

Makes the holder directly go to the new position by moving all axes simultaneously. Keyword args can be x,y,z,a or b.

Parameters
speed (*float*) – fraction of the standard speed setting (max 1.0)

property holder

The current specimen holder type.

property limits

Returns a dict with stage move limits.

move_to(***kwargs*)

Makes the holder safely move to the new position. Keyword args can be x,y,z,a or b.

property position

The current position of the stage (x,y,z in um and a,b in degrees).

property status

The current state of the stage.

class pytemscript.microscope.**Stem**(*microscope*)

STEM functions.

disable()

Switch back to TEM mode.

enable()

Switch to STEM mode.

property is_available

Returns whether the microscope has a STEM system or not.

property magnification

The magnification value in STEM mode. (read/write)

property rotation

The STEM rotation angle (in mrad). (read/write)

property scan_field_of_view

STEM full scan field of view. (read/write)

class pytemscript.microscope.**Temperature**(*microscope*)

LN dewars and temperature controls.

dewar_level(*dewar*)

Returns the LN level (%) in a dewar.

Parameters
dewar (*IntEnum*) – Dewar name (RefrigerantDewar enum)

property dewars_time

Returns remaining time (seconds) until the next dewar refill. Returns -1 if no refill is scheduled (e.g. All room temperature, or no dewar present).

force_refill()

Forces LN refill if the level is below 70%, otherwise returns an error. Note: this function takes considerable time to execute.

property is_available

Status of the temperature control. Should be always False on Tecnai instruments.

property is_dewar_filling

Returns TRUE if any of the dewars is currently busy filling.

property temp_cartridge

Returns Cartridge gripper temperature in Kelvins.

property temp_cassette

Returns Cassette gripper temperature in Kelvins.

property temp_docker

Returns Docker temperature in Kelvins.

property temp_holder

Returns Holder temperature in Kelvins.

class pytemscript.microscope.**UserDoor**(*microscope*)

User door hatch controls.

close()

Close the door.

open()

Open the door.

property state

Returns door state.

class pytemscript.microscope.**Vacuum**(*microscope*)

Vacuum functions.

column_close()

Close column valves.

column_open()

Open column valves.

property gauges

Returns a dict with vacuum gauges information. Pressure values are in Pascals.

property is_buffer_running

Checks whether the prevacuum pump is currently running (consequences: vibrations, exposure function blocked or should not be called).

property is_column_open

The status of the column valves.

run_buffer_cycle()

Runs a pumping cycle to empty the buffer.

property status

Status of the vacuum system.

1.3 Enumerations

Enumerations are represented with `IntEnum` objects and are used to describe TEM Scripting constants. When a property returns an enumeration, it will print its **name**. When you assign a variable to an enumeration, it will use its integer **value**. To see the accepted values, you can switch to source code using links below.

Example:

```
from pytemscript.microscope import Microscope
from pytemscript.utils.enums import *

microscope = Microscope()
stage = microscope.stage
print(stage.status)
'READY'  # <-- enumeration name

camera_size = AcqImageSize.FULL  # <-- enumeration value
image = microscope.acquisition.acquire_tem_image("BM-Ceta",
```

(continues on next page)

(continued from previous page)

```
                                                    size=camera_size,
                                                    exp_time=0.5,
                                                    binning=2)
```

class pytemscript.utils.enums.**AcqExposureMode**(*value*)
Exposure mode.

class pytemscript.utils.enums.**AcqImageCorrection**(*value*)
Image correction: unprocessed or corrected (gain/bias).

class pytemscript.utils.enums.**AcqImageFileFormat**(*value*)
Image file format.

class pytemscript.utils.enums.**AcqImageSize**(*value*)
Image size.

class pytemscript.utils.enums.**AcqMode**(*value*)
An enumeration.

class pytemscript.utils.enums.**AcqShutterMode**(*value*)
Shutter mode.

class pytemscript.utils.enums.**AcqSpeed**(*value*)
An enumeration.

class pytemscript.utils.enums.**ApertureType**(*value*)
Aperture type.

class pytemscript.utils.enums.**CassetteSlotStatus**(*value*)
Cassette slot status.

class pytemscript.utils.enums.**CondenserLensSystem**(*value*)
Two or three-condenser lens system.

class pytemscript.utils.enums.**CondenserMode**(*value*)
Condenser mode: parallel or probe.

class pytemscript.utils.enums.**DarkFieldMode**(*value*)
Dark field mode.

class pytemscript.utils.enums.**FegFlashingType**(*value*)
Cold FEG flashing type.

class pytemscript.utils.enums.**FegState**(*value*)
FEG state.

class pytemscript.utils.enums.**GaugePressureLevel**(*value*)
Vacuum gauge pressure level.

class pytemscript.utils.enums.**GaugeStatus**(*value*)
Vacuum gauge status.

class pytemscript.utils.enums.**HatchState**(*value*)
User door hatch state.

class pytemscript.utils.enums.**HighTensionState**(*value*)
High Tension status.

class pytemscript.utils.enums.**IlluminationMode**(*value*)

Illumination mode: nanoprobe or microprobe.

class pytemscript.utils.enums.**IlluminationNormalization**(*value*)

Normalization modes for condenser / objective lenses.

class pytemscript.utils.enums.**ImagePixelType**(*value*)

Image type: uint, int or float.

class pytemscript.utils.enums.**InstrumentMode**(*value*)

TEM or STEM mode.

class pytemscript.utils.enums.**LDState**(*value*)

Low Dose state.

class pytemscript.utils.enums.**LDStatus**(*value*)

Low Dose status: on or off.

class pytemscript.utils.enums.**LensProg**(*value*)

TEM or EFTEM mode.

class pytemscript.utils.enums.**MeasurementUnitType**(*value*)

Stage measurement units.

class pytemscript.utils.enums.**MechanismId**(*value*)

Aperture name.

class pytemscript.utils.enums.**MechanismState**(*value*)

Aperture state.

class pytemscript.utils.enums.**PlateLabelDateFormat**(*value*)

Date format for film.

class pytemscript.utils.enums.**ProductFamily**(*value*)

Microscope product family.

class pytemscript.utils.enums.**ProjDetectorShiftMode**(*value*)

This property determines, whether the chosen DetectorShift is changed when the fluorescent screen is moved down.

class pytemscript.utils.enums.**ProjectionDetectorShift**(*value*)

Sets the extra shift that projects the image/diffraction pattern onto a detector.

class pytemscript.utils.enums.**ProjectionMode**(*value*)

Imaging or diffraction.

class pytemscript.utils.enums.**ProjectionNormalization**(*value*)

Normalization modes for objective/projector lenses.

class pytemscript.utils.enums.**ProjectionSubMode**(*value*)

Magnification range mode.

class pytemscript.utils.enums.**RefrigerantDewar**(*value*)

Nitrogen dewar.

class pytemscript.utils.enums.**ScreenPosition**(*value*)

Fluscreen position.

class pytemscript.utils.enums.**StageAxes**(*value*)

Stage axes.

class pytemscript.utils.enums.**StageHolderType**(*value*)

Specimen holder type.

class pytemscript.utils.enums.**StageStatus**(*value*)

Stage status.

class pytemscript.utils.enums.**TEMScriptingError**(*value*)

Scripting error codes.

class pytemscript.utils.enums.**VacuumStatus**(*value*)

Vacuum system status.

1.4 Restrictions

The restrictions listed here are issues with the scripting interface itself. *pytemscript* only provides Python bindings to this scripting interface, thus these issues also occur using *pytemscript*. As there is no public list of known issues with the scripting interfaces by FEI or Thermo Fisher Scientific themself, known issues are listed here for the user's reference.

- On microscopes with just standard scripting only devices which are selected in the Microscope User Interface are available.
- Changing the projection mode from IMAGING to DIFFRACTION and back again changes the magnification in imaging mode (Titan 1.1).
- `optics.projection.magnification` does not return the actual magnification, but always 0.0 (Titan 1.1)
- Setting the binning value for a CCD camera, changes the exposure time (Titan 1.1 with Gatan US1000 camera).
- Acquisition with changed exposure time with a CCD camera, are not always done with the new exposure time.
- `optics.illumination.intensity_limit` raises exception when queried (Titan 1.1).
- `stage.go_to()` fails if movement is performed along multiple axes with speed keyword specified (internally the GoToWithSpeed method if the COM interface fails for multiple axes, Titan 1.1)
- If during a specimen holder exchange no holder is selected (yet), querying `stage.holder` fails (Titan 1.1).

1.5 Changelog

1.5.1 Version 3.0

- New changes are distributed under GPLv3+
- Interface re-written using comtypes library
- Standard scripting interfaces updated to v1.9
- Added Advanced scripting interfaces v1.2
- Development and testing are performed on:
  - Tecnai Spirit (WinXP, Python 3.4)
  - Tecnai F20 (Win7, Python 3.8)

  - Tecnai Polara (WinXP, Python 3.4)
  - Glacios (Win10, Python 3.8)
  - Tundra (Win10, Python 3.11)
  - Titan Krios G1 (Win7, Python 3.6), G2, G3i (Win10, Python 3.8)

1.5.2 Version 2.0.0

- C++ adapter removed, COM interface now directly accessed using `ctypes`
- Raised required minimum Python version to 3.4 (dropped support of Python 2.X)
- More extensive documentation of the high level interfaces and the pytemscript server
- Documentation of known issues of the original scripting interface
- Support of the fluorescent screen
- Separation of STEM detectors and CCD cameras in high level interface
- Deprecation of the methods 'get_detectors', 'get_detector_param', 'set_detector_params', and 'get_optics_state' of 'Microscope' and related classes. See docs for further details.
- Deprecation of the property 'AcqParams' of 'STEMDetector'. See docs for further details.
- Deprecation of the use of 'speed' and 'method' keywords in position dictionary of the 'set_stage_position' method.
- Abstract base class for high level interface
- Test scripts
- More illumination related functions
- TEM/STEM mode control
- Several small improvements and fixes

1.5.3 Version 1.0.10

- Speed keyword added Stage.Goto / Microscope.set_stage_position
- A lot of properties added to Microscope API (DiffShift, ObjStig, CondStig, Projection Mode / SubMode, Magnification, Normalization)
- More properties returned by Microscope.get_optics_state
- Timeout for RemoteMicroscope
- Lots of fixes

1.5.4 Version 1.0.9

- Normalization methods in new interface.
- Projective system settings in new interface.

1.5.5 Version 1.0.7

Started new interface (with client/server support).

1.5.6 Version 1.0.5

- Small fixes
- Clarified license: 3-clause BSD
- Compatibility to Py3K and anaconda distribution

1.5.7 Version 1.0.3

- Fixed some small things

1.5.8 Version 1.0.2

- Renamed project to pytemscript.
- Created documentation.

1.5.9 Version 1.0.1

- Fixed memory leak related to safearray handling

1.5.10 Version 1.0.0

- Initial release

CHAPTER

TWO

INSTALLATION

Warning: The project is still in development phase, no beta version has been released yet. Installing from sources is recommended.

Requirements:

- python >= 3.4
- comtypes
- mrcfile
- numpy
- sphinx-rtd-theme (optional, only for building documentation)
- matplotlib (optional, only for running tests)

2.1 Installation from PyPI on Windows

This assumes you have connection to the internet.

Execute from the command line (assuming you have your Python interpreter in the path):

```
py -m pip install --upgrade pip
py -m pip install pytemscript
```

2.2 Offline-Installation from wheels file on Windows

This assumes you have downloaded the wheels file <downloaded-wheels-file>.whl for temscript and comtypes into the current folder.

Execute from the command line (assuming you have your Python interpreter in the path:

```
py -m pip install numpy comtypes mrcfile pytemscript --no-index --find-links .
```

If you want to install pytemscript from sources (you still need to download comtypes *.whl):

```
py -m pip install numpy comtypes mrcfile --no-index --find-links .
py -m pip install -e <source_directory>
```

CHAPTER

THREE

SUPPORTED FUNCTIONS OF THE COM INTERFACE

Relative to TEM V1.9 standard scripting adapter:

- Acquisition
- ApertureMechanismCollection (untested)
- AutoLoader
- BlankerShutter
- Camera
- Configuration
- Gun
- Gun1 (untested)
- Illumination
- InstrumentModeControl
- Projection
- Stage
- TemperatureControl
- UserButton(s) (no events handling)
- Vacuum

Relative to TEM V1.2 advanced scripting adapter:

- Acquisitions
- Autoloader
- EnergyFilter (untested)
- Phaseplate
- PiezoStage (untested)
- Source (untested)
- TemperatureControl
- UserDoorHatch (untested)

CHAPTER

FOUR

QUICK EXAMPLE

Execute this on the microscope PC (with `pytemscript` package installed) to create an instance of the local `Microscope` interface:

```
from pytemscript.microscope import Microscope
microscope = Microscope()
```

Show the current acceleration voltage:

```
microscope.gun.voltage
300.0
```

Move beam:

```
beam_pos = microscope.optics.illumination.beam_shift
print(beam_pos)
(0.0, 0.0)
new_beam_pos = beam_pos[0], beam_pos[1] + 1.02
microscope.optics.illumination.beam_shift = new_beam_pos
```

Take an image:

```
image = microscope.acquisition.acquire_tem_image("BM-Ceta",
                                                 size=AcqImageSize.FULL,  # <-- see␣
↪enumerations
                                                 exp_time=0.5,
                                                 binning=2)
image.save("img.mrc")
```

CHAPTER

FIVE

DISCLAIMER

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

p

INDEX

M

N

O

P

R

S

T

U

V

W

Z