
camacq Documentation

Release 0.7.5

Martin Hjelmare

Oct 15, 2023

CONTENTS

1	camacq package	3
1.1	Subpackages	3
1.2	Submodules	17
1.3	camacq.bootstrap module	17
1.4	camacq.config module	18
1.5	camacq.const module	19
1.6	camacq.control module	19
1.7	camacq.event module	22
1.8	camacq.exceptions module	23
1.9	camacq.image module	24
1.10	camacq.log module	25
1.11	camacq.util module	26
2	Indices and tables	27
	Python Module Index	29
	Index	31

- [Project source at GitHub](#)

Contents:

CAMACQ PACKAGE

Control microscope through client server program.

1.1 Subpackages

1.1.1 `camacq.data` package

Store non python files.

1.1.2 `camacq.helper` package

Helper functions for camacq.

`camacq.helper.ensure_dict(value)`

Convert None to empty dict.

`camacq.helper.get_module(package, module_name)`

Return a module from a package.

Parameters

- **package** (*str*) – The path to the package.
- **module_name** (*str*) – The name of the module.

`camacq.helper.has_at_least_one_key(*keys)`

Validate that at least one key exists.

`camacq.helper.register_signals(center)`

Register signal handlers.

async `camacq.helper.setup_one_module(center, config, module)`

Set up one module or package.

Returns

Return a task to set up the module or None.

Return type

`asyncio.Task`

Submodules

camacq.helper.template module

Handle templates.

`camacq.helper.template.get_env(center)`

Get the template environment.

`camacq.helper.template.make_template(center, data)`

Make templated data.

`camacq.helper.template.render_template(data, variables)`

Render templated data.

`camacq.helper.template.template_next_well_x(sample, plate_name, x_wells=12, y_wells=8)`

Return the next well x coordinate for the plate x, y format.

`camacq.helper.template.template_next_well_xy(sample, plate_name, x_wells=12, y_wells=8)`

Return the next not done well for the given plate x, y format.

`camacq.helper.template.template_next_well_y(sample, plate_name, x_wells=12, y_wells=8)`

Return the next well x coordinate for the plate x, y format.

1.1.3 camacq.plugins package

Handle plugins.

`camacq.plugins.get_plugins()`

Return a dict of plugin modules.

async `camacq.plugins.setup_module(center, config)`

Set up the plugins package.

Parameters

- **center** (*Center instance*) – The Center instance.
- **config** (*dict*) – The config dict.

Subpackages

camacq.plugins.api package

Microscope API specific modules.

class `camacq.plugins.api.Api`

Bases: `object`

Represent the microscope API.

property name

Return the name of the API.

async send(*command*, ***kwargs*)

Send a command to the microscope API.

Parameters

command (*str*) – The command to send.

async send_many(*commands*, ***kwargs*)

Send multiple commands to the microscope API.

Parameters

commands (*list*) – A list of commands to send.

async start_imaging()

Send a command to the microscope to start the imaging.

async stop_imaging()

Send a command to the microscope to stop the imaging.

class camacq.plugins.api.**CommandEvent**(*data=None*)

Bases: [Event](#)

An event received from the API.

Notify with this event when a command is received via API.

property command

Return the command string.

Type

str

data

event_type = 'command_event'

class camacq.plugins.api.**ImageEvent**(*data=None*)

Bases: [Event](#)

An event received from the API.

Notify with this event when an image is saved via API.

property channel_id

Return channel id of the image.

Type

int

data

event_type = 'image_event'

property field_x

Return x coordinate of the well of the image.

Type

int

property field_y

Return y coordinate of the well of the image.

Type
int

property path

Return absolute path to the image.

Type
str

property plate_name

Return plate name of the image.

Type
str

property well_x

Return x coordinate of the well of the image.

Type
int

property well_y

Return y coordinate of the well of the image.

Type
int

property z_slice_id

Return z index of the image.

Type
int

class camacq.plugins.api.**StartCommandEvent**(data=None)

Bases: [CommandEvent](#)

An event received from the API.

Notify with this event when imaging starts via API.

property command

Return the command string.

Type
str

data

event_type = 'start_command_event'

class camacq.plugins.api.**StopCommandEvent**(data=None)

Bases: [CommandEvent](#)

An event received from the API.

Notify with this event when imaging stops via API.

property command

Return the command string.

Type
str

data

event_type = 'stop_command_event'

`camacq.plugins.api.register_api(center, api)`

Register api.

async `camacq.plugins.api.setup_module(center, config)`

Set up the microscope API package.

Parameters

- **center** (*Center instance*) – The Center instance.
- **config** (*dict*) – The config dict.

`camacq.plugins.api.validate_commands(value)`

Validate a template string via JSON.

camacq.plugins.automations package

Handle automations.

class `camacq.plugins.automations.ActionSequence(center, actions)`

Bases: object

Represent a sequence of actions.

delay(*seconds*, *variables*, *waiting*)

Delay action sequence.

Parameters

- **seconds** (*float*) – A time interval to delay the pending action sequence.
- **variables** (*dict*) – A dict of template variables.

class `camacq.plugins.automations.Automation(center, name, attach_triggers, cond_func, action_sequence, enabled=True)`

Bases: object

Automation class.

disable()

Disable automation.

enable()

Enable automation.

async trigger(*variables*)

Run actions of this automation.

class `camacq.plugins.automations.TemplateAction(center, action_conf)`

Bases: object

Representation of an action with template data.

render(*variables*)

Render the template with the kwargs for the action.

`camacq.plugins.automations.make_checker(condition_type, checks)`

Return a function to check condition.

async `camacq.plugins.automations.setup_module(center, config)`

Set up automations package.

Parameters

- **center** (*Center instance*) – The Center instance.
- **config** (*dict*) – The config dict.

`camacq.plugins.automations.template_check(value)`

Check if a rendered template string equals true.

If value is not a string, return value as is.

Submodules

camacq.plugins.automations.event module

Handle event trigger in automations.

`camacq.plugins.automations.event.handle_trigger(center, config, trigger_func)`

Listen for events.

camacq.plugins.leica package

Leica microscope API specific modules.

class `camacq.plugins.leica.LeicaApi(center, config, client)`

Bases: [Api](#)

Represent the Leica API.

property name

Return the name of the API.

async receive(*replies*)

Receive replies from CAM server and fire an event per reply.

Parameters

replies (*list*) – A list of replies from the CAM server.

async send(*command, **kwargs*)

Send a command to the Leica API.

Parameters

command (*list of tuples or string*) – The command to send.

async send_many(*commands, **kwargs*)

Send multiple commands to the microscope API.

Parameters

commands (*list*) – A list of commands to send.

async start_imaging()

Send a command to the microscope to start the imaging.

async start_listen()

Receive from the microscope socket.

async stop_imaging()

Send a command to the microscope to stop the imaging.

class camacq.plugins.leica.**LeicaCommandEvent**(data=None)

Bases: [CommandEvent](#)

Leica CommandEvent class.

property command

Return the command string.

data

event_type = 'leica_command_event'

class camacq.plugins.leica.**LeicaImageEvent**(data=None)

Bases: [ImageEvent](#)

Leica ImageEvent class.

property channel_id

Return channel id of the image.

Type

int

data

event_type = 'leica_image_event'

property field_x

Return x coordinate of the well of the image.

Type

int

property field_y

Return y coordinate of the well of the image.

Type

int

property job_id

Return job id of the image.

Type

int

property path

Return absolute path to the image.

Type

str

property plate_name

Return plate name of the image.

Type

str

property well_x

Return x coordinate of the well of the image.

Type

int

property well_y

Return y coordinate of the well of the image.

Type

int

property z_slice_id

Return z index of the image.

Type

int

class camacq.plugins.leica.**LeicaStartCommandEvent**(data=None)

Bases: [StartCommandEvent](#), [LeicaCommandEvent](#)

Leica StartCommandEvent class.

property command

Return the command string.

data

event_type = 'leica_start_command_event'

class camacq.plugins.leica.**LeicaStopCommandEvent**(data=None)

Bases: [StopCommandEvent](#), [LeicaCommandEvent](#)

Leica StopCommandEvent class.

property command

Return the command string.

data

event_type = 'leica_stop_command_event'

async camacq.plugins.leica.**setup_module**(center, config)

Set up Leica api package.

Parameters

- **center** (*Center instance*) – The Center instance.
- **config** (*dict*) – The config dict.

Submodules

camacq.plugins.leica.command module

Handle commands.

`camacq.plugins.leica.command.cam_com(exp, wellu, wellv, fieldx, fieldy, dxcoord, dycoord)`

Add a field to the cam list.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.camstart_com(afjob=None, afrange=None, afsteps=None)`

Start the cam scan with selected AF job and AF settings.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.camstop_com()`

Stop the cam scan.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.del_com()`

Delete the cam list.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.enable_com(wellu, wellv, fieldx, fieldy, enable)`

Enable a field in a well.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.gain_com(exp, num, value)`

Change the pmt gain in a job.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.start()`

Start the scan.

Return a list with parts for the cam command.

`camacq.plugins.leica.command.stop()`

Stop the scan.

Return a list with parts for the cam command.

camacq.plugins.leica.helper module

Helper functions for Leica api.

`camacq.plugins.leica.helper.find_image_path(relpath, root)`

Parse the relpath from the server to find file path from root.

Convert from windows path to posix path.

Parameters

- **relpath** (*str*) – A relative path to the image.
- **root** (*str*) – Path to directory where path should start.

Returns

Return path to image.

Return type

str

`camacq.plugins.leica.helper.get_field(path)`

Get path to field from image path.

Parameters

path (*string*) – Path to image.

Returns

Return path to field directory of image.

Return type

str

`camacq.plugins.leica.helper.get_imgs(path, img_type='tif', search='')`

Get all images below path.

Parameters

- **path** (*string*) – Path to directory where to search for images.
- **img_type** (*string*) – A string representing the image file type extension.
- **path** – A glob pattern string to use in the search.

Returns

Return paths of all images found.

Return type

list

`camacq.plugins.leica.helper.get_well(path)`

Get path to well from image path.

Parameters

path (*string*) – Path to image.

Returns

Return path to well directory of image.

Return type

str

camacq.plugins.sample package

Handle sample state.

class `camacq.plugins.sample.Image(path, values=None, **kwargs)`

Bases: *ImageContainer*

An image with path and position info.

property `change_event`

Return an event class to fire on container change.

Type

Event

property images

Return a dict with all images for the container.

Type
dict

property name

Return an identifying name for the container.

Type
str

property path

Return the path of the image.

Type
str

property values

Return a dict with the values set for the container.

Type
dict

class camacq.plugins.sample.**ImageContainer**

Bases: ABC

A container for images.

abstract property change_event

Return an event class to fire on container change.

Type
Event

abstract property images

Return a dict with all images for the container.

Type
dict

abstract property name

Return an identifying name for the container.

Type
str

abstract property values

Return a dict with the values set for the container.

Type
dict

class camacq.plugins.sample.**Sample**

Bases: *ImageContainer*, ABC

Representation of the state of the sample.

center = None

abstract property change_event

Return an event class to fire on container change.

Type

Event

data = None

get_sample(*name*, ***kwargs*)

Get an image container of the sample.

Parameters

- **name** (*str*) – The name of the container type.
- ****kwargs** – Arbitrary keyword arguments. These will be used to create the id string of the container.

Returns

Return the found ImageContainer instance.

Return type

ImageContainer instance

abstract property image_event_type

Return the image event type to listen to for the sample.

Type

str

abstract property images

Return a dict with all images for the container.

Type

dict

abstract property name

Return the name of the sample.

Type

str

abstract async on_image(*center*, *event*)

Handle image event for this sample.

async set_sample(*name*, *values=None*, ***kwargs*)

Set an image container of the sample.

Parameters

- **name** (*str*) – The name of the container type.
- **values** (*dict*) – The optional values to set on the container.
- ****kwargs** – Arbitrary keyword arguments. These will be used to create the id string of the container.

Returns

Return the ImageContainer instance that was updated.

Return type

ImageContainer instance

abstract property set_sample_schema

Return the validation schema of the set_sample method.

abstract property values

Return a dict with the values set for the container.

Type
dict

class camacq.plugins.sample.**SampleEvent**(data=None)

Bases: [Event](#)

An event produced by a sample change event.

property container

Return the container instance of the event.

Type
ImageContainer instance

property container_name

Return the container name of the event.

Type
str

data

event_type = 'sample_event'

property images

Return the container images of the event.

Type
dict

property values

Return the container values of the event.

Type
dict

class camacq.plugins.sample.**SampleImageSetEvent**(data=None)

Bases: [SampleEvent](#)

An event produced by a new image on the sample.

property container

Return the container instance of the event.

Type
ImageContainer instance

property container_name

Return the container name of the event.

Type
str

data

event_type = 'sample_image_set_event'

property images

Return the container images of the event.

Type
dict

property values

Return the container values of the event.

Type
dict

class camacq.plugins.sample.Samples

Bases: *dotdict*

Hold all samples.

clear() → None. Remove all items from D.

copy() → a shallow copy of D

fromkeys(value=None, /)

Create a new dictionary with keys from iterable and values set to value.

get(key, default=None, /)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

pop(k[, d]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

setdefault(key, default=None, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update([E], **F) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values() → an object providing a view on D's values

camacq.plugins.sample.**get_matched_samples**(sample, name, attrs=None, values=None)

Return the sample items that match.

camacq.plugins.sample.**register_sample**(center, sample)

Register sample.

async `camacq.plugins.sample.setup_module(center, config)`

Set up sample module.

Parameters

- **center** (*Center instance*) – The Center instance.
- **config** (*dict*) – The config dict.

Submodules

camacq.plugins.sample.helper module

Submodules

camacq.plugins.rename_image module

Handle renaming of an image.

`camacq.plugins.rename_image.rename_image(old_path, new_path)`

Rename image at *old_path* to *new_path*.

Parameters

- **old_path** (*str*) – The absolute path to the existing image.
- **new_path** (*str*) – The absolute path to the renamed image.

async `camacq.plugins.rename_image.setup_module(center, config)`

Set up image rename plugin.

Parameters

- **center** (*Center instance*) – The Center instance.
- **config** (*dict*) – The config dict.

1.2 Submodules

1.3 camacq.bootstrap module

Configure and set up control center.

async `camacq.bootstrap.setup_dict(center, config)`

Set up control center from config dict.

Parameters

config (*dict*) – The config dict.

Returns

Return the Center instance.

Return type

Center instance

async `camacq.bootstrap.setup_file(config_file, cmd_args)`

Set up control center from config file and command line args.

Parameters

- **config_file** (*str*) – The path to the configuration YAML file.
- **cmd_args** (*dict*) – The dict with the command line arguments.

Returns

Return the Center instance.

Return type

Center instance

1.4 camacq.config module

Handle the config file.

`camacq.config.create_default_config(config_dir)`

Create a default config file in given configuration directory.

Parameters

config_dir (*pathlib.Path*) – The path to the configuration directory.

Returns

Return path to new configuration file if success, None if failed.

Return type

pathlib.Path

`camacq.config.ensure_config_exists(config_dir)`

Ensure configuration file exists in the configuration directory.

Create a default configuration file if needed.

Parameters

config_dir (*pathlib.Path*) – The path to the configuration directory.

Returns

Return path to the configuration file.

Return type

pathlib.Path

`camacq.config.find_config_file(config_dir)`

Find the configuration file in the configuration directory.

Parameters

config_dir (*pathlib.Path*) – The path to the configuration directory.

Returns

Return path to the configuration file if found, None if not found.

Return type

pathlib.Path

`camacq.config.get_default_config_dir()`

Get the default configuration directory based on OS.

Returns

Return the path to the configuration directory.

Return type

str

`camacq.config.load_config_file(path)`

Parse a YAML configuration file.

Parameters

path (*pathlib.Path*) – The path to the configuration YAML file.

Returns

Return a dict with the configuration contents.

Return type

dict

1.5 camacq.const module

Store common constants.

1.6 camacq.control module

Control the microscope.

class `camacq.control.Action(action_type, action_id, func, schema)`

Bases: `object`

Represent an action.

class `camacq.control.ActionType`

Bases: `dotdict`

Represent an action type.

clear() → None. Remove all items from D.

copy() → a shallow copy of D

fromkeys(value=None, /)

Create a new dictionary with keys from iterable and values set to value.

get(key, default=None, /)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

pop(k[, d]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises `KeyError` if the dict is empty.

setdefault(*key*, *default=None*, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update([*E*], ***F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values() → an object providing a view on D's values

class camacq.control.ActionsRegistry(*center*)

Bases: object

Manage all registered actions.

property actions

Return dict of ActionTypes with all registered actions.

Type
dict

async call(*action_type*, *action_id*, ***kwargs*)

Call an action with optional kwargs.

Parameters

- **action_type** (*str*) – The name of the module where the action is registered.
- **action_id** (*str*) – The id of the action to call.
- ****kwargs** – Arbitrary keyword arguments. These will be passed to the action function when an action is called.

register(*action_type*, *action_id*, *action_func*, *schema*)

Register an action.

Register actions per module.

Parameters

- **action_type** (*str*) – The name of the action_type to register the action under.
- **action_id** (*str*) – The id of the action to register.
- **action_func** (*voluptuous schema*) – The function that should be called for the action.
- **action_func** – The voluptuous schema that should validate the parameters of the action call.

class camacq.control.CamAcqStartEvent(*data=None*)

Bases: [Event](#)

An event fired when camacq has started.

data

event_type = 'camacq_start_event'

class camacq.control.CamAcqStopEvent(*data=None*)

Bases: [Event](#)

An event fired when camacq is about to stop.

data

event_type = 'camacq_stop_event'

property exit_code

Return the plate instance of the event.

Type

int

class camacq.control.**Center**(*loop=None*)

Bases: object

Represent a control center for the microscope.

Parameters

loop (*asyncio.EventLoop*) – The event loop.

loop

Return the event loop.

Type

asyncio.EventLoop

bus

Return the EventBus instance.

Type

EventBus instance

samples

Return the Samples instance that holds all the Sample instances.

Type

Samples instance

actions

Return the ActionsRegistry instance.

Type

ActionsRegistry instance

data

Return dict that stores data from other modules than control.

Type

dict

add_executor_job(*func, *args*)

Schedule a function to be run in the thread pool.

Return a task.

create_task(*coro*)

Schedule a coroutine on the event loop.

Return a task.

async end(*code*)

Prepare app for exit.

Parameters

code (*int*) – Exit code to return when the app exits.

async start()

Start the app.

async wait_for()

Wait for all pending tasks.

camacq.control.loop_exception_handler(*loop, context*)

Handle exceptions inside the event loop.

1.7 camacq.event module

Hold events.

class camacq.event.**Event**(*data=None*)

Bases: object

A base event.

Parameters

data (*dict*) – The data of the event.

data

Return the data of the event.

Type

dict

data

event_type = 'base_event'

class camacq.event.**EventBus**(*center*)

Bases: object

Representation of an eventbus.

Parameters

center (*Center instance*) – The Center instance.

property event_types

Return all registered event types.

Type

list

async notify(*event*)

Notify handlers that an event has fired.

Parameters

event (*Event instance*) – An instance of Event or an instance of subclass of Event.

register(*event_type, handler*)

Register event handler and return a function to remove it.

An event can be a message from the microscope API or an internal event.

Parameters

- **event_type** (*str*) – A string representing the type of event.
- **handler** (*callable*) – A coroutine function that should accept two parameters, center and event. The first parameter is the Center instance, the second parameter is the Event instance that has fired.

Returns

Return a function to remove the registered handler.

Return type

callable

`camacq.event.match_event(event, **event_data)`

Return True if event attributes match event_data.

1.8 camacq.exceptions module

Provide exceptions.

exception `camacq.exceptions.CamAcqError`

Bases: `Exception`

Represent the base camacq exception.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `camacq.exceptions.MissingActionError(action_id)`

Bases: `CamAcqError`

Represent a missing action error.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `camacq.exceptions.MissingActionTypeError(action_type)`

Bases: `CamAcqError`

Represent a missing action type error.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `camacq.exceptions.SampleError`

Bases: `CamAcqError`

Represent a sample error.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `camacq.exceptions.TemplateError(exc)`

Bases: `CamAcqError`

Represent a template error.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

1.9 camacq.image module

Handle images.

class `camacq.image.ImageData(path=None, data=None, metadata=None)`

Bases: `object`

Represent the data of an image with path, data, metadata and histogram.

Parameters

- **path** (*str*) – Path to the image.
- **data** (*numpy array*) – A numpy array with the image data.
- **metadata** (*dict*) – The meta data of the image as a JSON dict.

path

The path to the image.

Type

`str`

property data

Return the data of the image.

Setter

Set the data of the image.

Type

`numpy array`

property histogram

Calculate and return image histogram.

Type

`numpy array`

property metadata

Return metadata of image.

Setter

Set the meta data of the image.

Type

`str`

save(*path=None, data=None, metadata=None*)

Save image with image data and optional meta data.

Parameters

- **path** (*str*) – The path to the image.
- **data** (*numpy array*) – A numpy array with the image data.
- **metadata** (*dict*) – The meta data of the image as a JSON dict.

camacq.image.make_proj(*images*)

Make a dict of max projections from a dict of channels and paths.

Each channel will make one max projection.

Parameters

images (*dict*) – Dict of paths and channel ids.

Returns

Return a dict of channels that map ImageData objects. Each image object have a max projection as data.

Return type

dict

camacq.image.read_image(*path*)

Read a tif image and return the data.

Parameters

path (*str*) – The path to the image.

Returns

Return a numpy array with image data.

Return type

numpy array

camacq.image.save_image(*path, data, description=None*)

Save a tif image with image data and meta data.

Parameters

- **path** (*str*) – The path to the image.
- **data** (*numpy array*) – A numpy array with the image data.
- **description** (*str*) – The description string of the image.

1.10 camacq.log module

Handle logging.

camacq.log.check_path(*path*)

Check that path to config exists and is writable for logging.

Parameters

path (*pathlib.Path*) – The path to the log file or log directory.

Returns

Return True if path exists and is writable.

Return type

bool

`camacq.log.enable_log(config)`

Enable logging.

Parameters**config** (*dict*) – The dict with the configuration.

1.11 camacq.util module

Host utils that are not aware of the implementation of camacq.

class `camacq.util.dotdict`

Bases: dict

Access to dictionary attributes with dot notation.

clear() → None. Remove all items from D.**copy**() → a shallow copy of D**fromkeys**(*value=None, /*)

Create a new dictionary with keys from iterable and values set to value.

get(*key, default=None, /*)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items**keys**() → a set-like object providing a view on D's keys**pop**(*k[, d]*) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

setdefault(*key, default=None, /*)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update(*[E], **F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values() → an object providing a view on D's values

INDICES AND TABLES

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

PYTHON MODULE INDEX

C

- camacq, 3
- camacq.bootstrap, 17
- camacq.config, 18
- camacq.const, 19
- camacq.control, 19
- camacq.data, 3
- camacq.event, 22
- camacq.exceptions, 23
- camacq.helper, 3
- camacq.helper.template, 4
- camacq.image, 24
- camacq.log, 25
- camacq.plugins, 4
- camacq.plugins.api, 4
- camacq.plugins.automations, 7
- camacq.plugins.automations.event, 8
- camacq.plugins.leica, 8
- camacq.plugins.leica.command, 11
- camacq.plugins.leica.helper, 11
- camacq.plugins.rename_image, 17
- camacq.plugins.sample, 12
- camacq.util, 26

A

Action (class in *camacq.control*), 19
actions (*camacq.control.ActionsRegistry* property), 20
actions (*camacq.control.Center* attribute), 21
ActionSequence (class in *camacq.plugins.automations*), 7
ActionsRegistry (class in *camacq.control*), 20
ActionType (class in *camacq.control*), 19
add_executor_job() (*camacq.control.Center* method), 21
Api (class in *camacq.plugins.api*), 4
args (*camacq.exceptions.CamAcqError* attribute), 23
args (*camacq.exceptions.MissingActionError* attribute), 23
args (*camacq.exceptions.MissingActionTypeError* attribute), 23
args (*camacq.exceptions.SampleError* attribute), 23
args (*camacq.exceptions.TemplateError* attribute), 24
Automation (class in *camacq.plugins.automations*), 7

B

bus (*camacq.control.Center* attribute), 21

C

call() (*camacq.control.ActionsRegistry* method), 20
cam_com() (in module *camacq.plugins.leica.command*), 11
camacq
 module, 3
camacq.bootstrap
 module, 17
camacq.config
 module, 18
camacq.const
 module, 19
camacq.control
 module, 19
camacq.data
 module, 3
camacq.event
 module, 22
camacq.exceptions

 module, 23
camacq.helper
 module, 3
camacq.helper.template
 module, 4
camacq.image
 module, 24
camacq.log
 module, 25
camacq.plugins
 module, 4
camacq.plugins.api
 module, 4
camacq.plugins.automations
 module, 7
camacq.plugins.automations.event
 module, 8
camacq.plugins.leica
 module, 8
camacq.plugins.leica.command
 module, 11
camacq.plugins.leica.helper
 module, 11
camacq.plugins.rename_image
 module, 17
camacq.plugins.sample
 module, 12
camacq.util
 module, 26
CamAcqError, 23
CamAcqStartEvent (class in *camacq.control*), 20
CamAcqStopEvent (class in *camacq.control*), 20
camstart_com() (in module *camacq.plugins.leica.command*), 11
camstop_com() (in module *camacq.plugins.leica.command*), 11
center (*camacq.plugins.sample.Sample* attribute), 13
Center (class in *camacq.control*), 21
change_event (*camacq.plugins.sample.Image* property), 12
change_event (*camacq.plugins.sample.ImageContainer* property), 13

`change_event` (*camacq.plugins.sample.Sample* property), 13
`channel_id` (*camacq.plugins.api.ImageEvent* property), 5
`channel_id` (*camacq.plugins.leica.LeicaImageEvent* property), 9
`check_path()` (in module *camacq.log*), 25
`clear()` (*camacq.control.ActionType* method), 19
`clear()` (*camacq.plugins.sample.Samples* method), 16
`clear()` (*camacq.util.dotdict* method), 26
`command` (*camacq.plugins.api.CommandEvent* property), 5
`command` (*camacq.plugins.api.StartCommandEvent* property), 6
`command` (*camacq.plugins.api.StopCommandEvent* property), 6
`command` (*camacq.plugins.leica.LeicaCommandEvent* property), 9
`command` (*camacq.plugins.leica.LeicaStartCommandEvent* property), 10
`command` (*camacq.plugins.leica.LeicaStopCommandEvent* property), 10
`CommandEvent` (class in *camacq.plugins.api*), 5
`container` (*camacq.plugins.sample.SampleEvent* property), 15
`container` (*camacq.plugins.sample.SampleImageSetEvent* property), 15
`container_name` (*camacq.plugins.sample.SampleEvent* property), 15
`container_name` (*camacq.plugins.sample.SampleImageSetEvent* property), 15
`copy()` (*camacq.control.ActionType* method), 19
`copy()` (*camacq.plugins.sample.Samples* method), 16
`copy()` (*camacq.util.dotdict* method), 26
`create_default_config()` (in module *camacq.config*), 18
`create_task()` (*camacq.control.Center* method), 21

D

`data` (*camacq.control.CamAcqStartEvent* attribute), 20
`data` (*camacq.control.CamAcqStopEvent* attribute), 20
`data` (*camacq.control.Center* attribute), 21
`data` (*camacq.event.Event* attribute), 22
`data` (*camacq.image.ImageData* property), 24
`data` (*camacq.plugins.api.CommandEvent* attribute), 5
`data` (*camacq.plugins.api.ImageEvent* attribute), 5
`data` (*camacq.plugins.api.StartCommandEvent* attribute), 6
`data` (*camacq.plugins.api.StopCommandEvent* attribute), 6
`data` (*camacq.plugins.leica.LeicaCommandEvent* attribute), 9
`data` (*camacq.plugins.leica.LeicaImageEvent* attribute), 9

`data` (*camacq.plugins.leica.LeicaStartCommandEvent* attribute), 10
`data` (*camacq.plugins.leica.LeicaStopCommandEvent* attribute), 10
`data` (*camacq.plugins.sample.Sample* attribute), 14
`data` (*camacq.plugins.sample.SampleEvent* attribute), 15
`data` (*camacq.plugins.sample.SampleImageSetEvent* attribute), 15
`del_com()` (in module *camacq.plugins.leica.command*), 11
`delay()` (*camacq.plugins.automations.ActionSequence* method), 7
`disable()` (*camacq.plugins.automations.Automation* method), 7
`dotdict` (class in *camacq.util*), 26

E

`enable()` (*camacq.plugins.automations.Automation* method), 7
`enable_com()` (in module *camacq.plugins.leica.command*), 11
`enable_log()` (in module *camacq.log*), 26
`end()` (*camacq.control.Center* method), 21
`ensure_config_exists()` (in module *camacq.config*), 18
`ensure_dict()` (in module *camacq.helper*), 3
`Event` (class in *camacq.event*), 22
`event_type` (*camacq.control.CamAcqStartEvent* attribute), 20
`event_type` (*camacq.control.CamAcqStopEvent* attribute), 21
`event_type` (*camacq.event.Event* attribute), 22
`event_type` (*camacq.plugins.api.CommandEvent* attribute), 5
`event_type` (*camacq.plugins.api.ImageEvent* attribute), 5
`event_type` (*camacq.plugins.api.StartCommandEvent* attribute), 6
`event_type` (*camacq.plugins.api.StopCommandEvent* attribute), 7
`event_type` (*camacq.plugins.leica.LeicaCommandEvent* attribute), 9
`event_type` (*camacq.plugins.leica.LeicaImageEvent* attribute), 9
`event_type` (*camacq.plugins.leica.LeicaStartCommandEvent* attribute), 10
`event_type` (*camacq.plugins.leica.LeicaStopCommandEvent* attribute), 10
`event_type` (*camacq.plugins.sample.SampleEvent* attribute), 15
`event_type` (*camacq.plugins.sample.SampleImageSetEvent* attribute), 15
`event_types` (*camacq.event.EventBus* property), 22
`EventBus` (class in *camacq.event*), 22

`exit_code` (*camacq.control.CamAcqStopEvent* property), 21

F

`field_x` (*camacq.plugins.api.ImageEvent* property), 5
`field_x` (*camacq.plugins.leica.LeicaImageEvent* property), 9
`field_y` (*camacq.plugins.api.ImageEvent* property), 5
`field_y` (*camacq.plugins.leica.LeicaImageEvent* property), 9
`find_config_file()` (in module *camacq.config*), 18
`find_image_path()` (in module *camacq.plugins.leica.helper*), 11
`fromkeys()` (*camacq.control.ActionType* method), 19
`fromkeys()` (*camacq.plugins.sample.Samples* method), 16
`fromkeys()` (*camacq.util.dotdict* method), 26

G

`gain_com()` (in module *camacq.plugins.leica.command*), 11
`get()` (*camacq.control.ActionType* method), 19
`get()` (*camacq.plugins.sample.Samples* method), 16
`get()` (*camacq.util.dotdict* method), 26
`get_default_config_dir()` (in module *camacq.config*), 18
`get_env()` (in module *camacq.helper.template*), 4
`get_field()` (in module *camacq.plugins.leica.helper*), 12
`get_imgs()` (in module *camacq.plugins.leica.helper*), 12
`get_matched_samples()` (in module *camacq.plugins.sample*), 16
`get_module()` (in module *camacq.helper*), 3
`get_plugins()` (in module *camacq.plugins*), 4
`get_sample()` (*camacq.plugins.sample.Sample* method), 14
`get_well()` (in module *camacq.plugins.leica.helper*), 12

H

`handle_trigger()` (in module *camacq.plugins.automations.event*), 8
`has_at_least_one_key()` (in module *camacq.helper*), 3
`histogram` (*camacq.image.ImageData* property), 24

I

`Image` (class in *camacq.plugins.sample*), 12
`image_event_type` (*camacq.plugins.sample.Sample* property), 14
`ImageContainer` (class in *camacq.plugins.sample*), 13
`ImageData` (class in *camacq.image*), 24
`ImageEvent` (class in *camacq.plugins.api*), 5
`images` (*camacq.plugins.sample.Image* property), 12

`images` (*camacq.plugins.sample.ImageContainer* property), 13
`images` (*camacq.plugins.sample.Sample* property), 14
`images` (*camacq.plugins.sample.SampleEvent* property), 15
`images` (*camacq.plugins.sample.SampleImageSetEvent* property), 16
`items()` (*camacq.control.ActionType* method), 19
`items()` (*camacq.plugins.sample.Samples* method), 16
`items()` (*camacq.util.dotdict* method), 26

J

`job_id` (*camacq.plugins.leica.LeicaImageEvent* property), 9

K

`keys()` (*camacq.control.ActionType* method), 19
`keys()` (*camacq.plugins.sample.Samples* method), 16
`keys()` (*camacq.util.dotdict* method), 26

L

`LeicaApi` (class in *camacq.plugins.leica*), 8
`LeicaCommandEvent` (class in *camacq.plugins.leica*), 9
`LeicaImageEvent` (class in *camacq.plugins.leica*), 9
`LeicaStartCommandEvent` (class in *camacq.plugins.leica*), 10
`LeicaStopCommandEvent` (class in *camacq.plugins.leica*), 10
`load_config_file()` (in module *camacq.config*), 19
`loop` (*camacq.control.Center* attribute), 21
`loop_exception_handler()` (in module *camacq.control*), 22

M

`make_checker()` (in module *camacq.plugins.automations*), 7
`make_proj()` (in module *camacq.image*), 25
`make_template()` (in module *camacq.helper.template*), 4
`match_event()` (in module *camacq.event*), 23
`metadata` (*camacq.image.ImageData* property), 24
`MissingActionError`, 23
`MissingActionTypeError`, 23
`module`
 camacq, 3
 camacq.bootstrap, 17
 camacq.config, 18
 camacq.const, 19
 camacq.control, 19
 camacq.data, 3
 camacq.event, 22
 camacq.exceptions, 23
 camacq.helper, 3

camacq.helper.template, 4
camacq.image, 24
camacq.log, 25
camacq.plugins, 4
camacq.plugins.api, 4
camacq.plugins.automations, 7
camacq.plugins.automations.event, 8
camacq.plugins.leica, 8
camacq.plugins.leica.command, 11
camacq.plugins.leica.helper, 11
camacq.plugins.rename_image, 17
camacq.plugins.sample, 12
camacq.util, 26

N

name (camacq.plugins.api.Api property), 4
name (camacq.plugins.leica.LeicaApi property), 8
name (camacq.plugins.sample.Image property), 13
name (camacq.plugins.sample.ImageContainer property), 13
name (camacq.plugins.sample.Sample property), 14
notify() (camacq.event.EventBus method), 22

O

on_image() (camacq.plugins.sample.Sample method), 14

P

path (camacq.image.ImageData attribute), 24
path (camacq.plugins.api.ImageEvent property), 6
path (camacq.plugins.leica.LeicaImageEvent property), 9
path (camacq.plugins.sample.Image property), 13
plate_name (camacq.plugins.api.ImageEvent property), 6
plate_name (camacq.plugins.leica.LeicaImageEvent property), 9
pop() (camacq.control.ActionType method), 19
pop() (camacq.plugins.sample.Samples method), 16
pop() (camacq.util.dotdict method), 26
popitem() (camacq.control.ActionType method), 19
popitem() (camacq.plugins.sample.Samples method), 16
popitem() (camacq.util.dotdict method), 26

R

read_image() (in module camacq.image), 25
receive() (camacq.plugins.leica.LeicaApi method), 8
register() (camacq.control.ActionsRegistry method), 20
register() (camacq.event.EventBus method), 22
register_api() (in module camacq.plugins.api), 7
register_sample() (in module camacq.plugins.sample), 16

register_signals() (in module camacq.helper), 3
rename_image() (in module camacq.plugins.rename_image), 17
render() (camacq.plugins.automations.TemplateAction method), 7
render_template() (in module camacq.helper.template), 4

S

Sample (class in camacq.plugins.sample), 13
SampleError, 23
SampleEvent (class in camacq.plugins.sample), 15
SampleImageSetEvent (class in camacq.plugins.sample), 15
samples (camacq.control.Center attribute), 21
Samples (class in camacq.plugins.sample), 16
save() (camacq.image.ImageData method), 24
save_image() (in module camacq.image), 25
send() (camacq.plugins.api.Api method), 4
send() (camacq.plugins.leica.LeicaApi method), 8
send_many() (camacq.plugins.api.Api method), 5
send_many() (camacq.plugins.leica.LeicaApi method), 8
set_sample() (camacq.plugins.sample.Sample method), 14
set_sample_schema (camacq.plugins.sample.Sample property), 14
setdefault() (camacq.control.ActionType method), 19
setdefault() (camacq.plugins.sample.Samples method), 16
setdefault() (camacq.util.dotdict method), 26
setup_dict() (in module camacq.bootstrap), 17
setup_file() (in module camacq.bootstrap), 17
setup_module() (in module camacq.plugins), 4
setup_module() (in module camacq.plugins.api), 7
setup_module() (in module camacq.plugins.automations), 8
setup_module() (in module camacq.plugins.leica), 10
setup_module() (in module camacq.plugins.rename_image), 17
setup_module() (in module camacq.plugins.sample), 16
setup_one_module() (in module camacq.helper), 3
start() (camacq.control.Center method), 22
start() (in module camacq.plugins.leica.command), 11
start_imaging() (camacq.plugins.api.Api method), 5
start_imaging() (camacq.plugins.leica.LeicaApi method), 8
start_listen() (camacq.plugins.leica.LeicaApi method), 9
StartCommandEvent (class in camacq.plugins.api), 6
stop() (in module camacq.plugins.leica.command), 11
stop_imaging() (camacq.plugins.api.Api method), 5

`stop_imaging()` (*camacq.plugins.leica.LeicaApi* method), 9
`StopCommandEvent` (class in *camacq.plugins.api*), 6

T

`template_check()` (in module *camacq.plugins.automations*), 8
`template_next_well_x()` (in module *camacq.helper.template*), 4
`template_next_well_xy()` (in module *camacq.helper.template*), 4
`template_next_well_y()` (in module *camacq.helper.template*), 4
`TemplateAction` (class in *camacq.plugins.automations*), 7
`TemplateError`, 23
`trigger()` (*camacq.plugins.automations.Automation* method), 7

U

`update()` (*camacq.control.ActionType* method), 20
`update()` (*camacq.plugins.sample.Samples* method), 16
`update()` (*camacq.util.dotdict* method), 26

V

`validate_commands()` (in module *camacq.plugins.api*), 7
`values` (*camacq.plugins.sample.Image* property), 13
`values` (*camacq.plugins.sample.ImageContainer* property), 13
`values` (*camacq.plugins.sample.Sample* property), 15
`values` (*camacq.plugins.sample.SampleEvent* property), 15
`values` (*camacq.plugins.sample.SampleImageSetEvent* property), 16
`values()` (*camacq.control.ActionType* method), 20
`values()` (*camacq.plugins.sample.Samples* method), 16
`values()` (*camacq.util.dotdict* method), 26

W

`wait_for()` (*camacq.control.Center* method), 22
`well_x` (*camacq.plugins.api.ImageEvent* property), 6
`well_x` (*camacq.plugins.leica.LeicalImageEvent* property), 10
`well_y` (*camacq.plugins.api.ImageEvent* property), 6
`well_y` (*camacq.plugins.leica.LeicalImageEvent* property), 10
`with_traceback()` (*camacq.exceptions.CamAcqError* method), 23
`with_traceback()` (*camacq.exceptions.MissingActionError* method), 23

`with_traceback()` (*camacq.exceptions.MissingActionTypeError* method), 23

`with_traceback()` (*camacq.exceptions.SampleError* method), 23

`with_traceback()` (*camacq.exceptions.TemplateError* method), 24

Z

`z_slice_id` (*camacq.plugins.api.ImageEvent* property), 6

`z_slice_id` (*camacq.plugins.leica.LeicalImageEvent* property), 10