moco-wrapper Documentation

Release 0.9.0

sommalia

Dec 24, 2020

Getting Started

This documenation is split into the following parts:

CHAPTER 1

Quick Start

First step is installing the package.

```
$ pip install moco-wrapper
```

After that we need two things. The moco domain and the user api key.

Getting your moco domain is the easy part. From your full url just take your company specific part. For example if your moco appkication is found under https://testabcd.mocoapp.com we just use **testabcd** for creating our object.

Next we need an api key. You can find this key when logging into your user account, going to your user profile and copying the value under *your personal api key* from the tab integrations. (See https://github.com/hundertzehn/mocoapp-api-docs#authentication for more information)

Then we all the information we need to create the moco instance:

```
>> from moco_wrapper import Moco
>> wrapper = Moco(auth={"domain": "testabcd", "api_key": "your api key"})
```

After that we make a test call to the api.

```
>> projects = wrapper.Project.getlist()
>> print(projects)
<PagedListResponse, Status Code: 200, Data: [<moco_wrapper.models.objector_models.
↪project.Project at ..]>
```

CHAPTER 2

Installation

The moco-wrapper package is hosted on pypi and can be installed via pip.

```
$ pip install moco-wrapper
```

Note: Depending on your system, you may need to use pip3 to install packages for Python 3.

Note: This package was only developed for Python 3. Using it with Python 2 will result in errors.

2.1 Installation in a virtual environment

```
$ apt-get install python3-venv
$ python3 -m venv venv
$ source venv/bin/activate
$ (venv) pip install moco-wrapper
```

2.2 Upgrading moco-wrapper

The moco-wrapper can be updated by using pip

```
$ pip install --upgrade moco-wrapper
```

2.3 Install from source

Warning: For installation from source we recommend you install it into a virtual environment.

```
$ sudo apt-get install python3 python3-venv git make
$ python3 -m venv venv
$ source venv/bin/activate
$ (venv) git clone https://github.com/sommalia/moco-wrapper moco_wrapper
$ (venv) cd moco_wrapper
$ (venv) make install
```

CHAPTER 3

Authentication

There are two ways of authenticating yourself with the moco api. Via api key and via your own user email and password.

3.1 Via api key

You can find your own api key under your user profile when you log into your account.

```
from moco_wrapper import Moco

wrapper = Moco(
    auth={
        "api_key": "[YOUR API KEY]"
        "domain": "testabcd" #your full domain would be testabcd.mocoapp.com
    }
)
```

Note: The api key is always associated with the user it belongs to. Things like activities and presences always work in context of the current user. If you want to be someone else see *Impersonation*.

Note: This method is faster than authenticating via username and password because it skips the authentication requests (with an api key, you already are authenticated).

3.2 Via username and password

The second way you can authencate is via your own user information (username and password).

```
from moco_wrapper import Moco

wrapper = Moco(
    auth={
        "email": "my-account-email@mycomapany.com",
        "password": "my account password",
        "domain": "testabcd" #full domain is testabcd.mocoapp.com
    }
)
```

Note: If you authenticate in this way an extra request will be sent, before you try to request any actual ressources of the api, for obtaining valid authentication.

CHAPTER 4

Impersonation

Things like *moco_wrapper.models.Activity* and *moco_wrapper.models.UserPresence* are always mapped to the user of the api key that was used.

If you want to change that (for example when logging activities on behalf of someone else), you can impersontate them.

Note: For impersonating other users your own user account must have the `Staff` permission. See https://github.com/hundertzehn/mocoapp-api-docs#impersonation.

You can start off the moco instance already impersonating a user:

```
from moco_wrapper import moco

impersonate_as = 43
m = Moco(
    ..,
    impersonate_user_id = impersonate_as
)

## do something as someone else
##
##

## reset impersonation
m.clear_impersonation()
```

Note: If you are authenticating yourself with email and password, that the first request (for authenticating you) will still take place as yourself, impersonation will only occur after you are already authenticated.

Or you can use the impersonte method by itself.

```
from moco_wrapper import Moco

m = Moco()

## do something as you
##
##

impersonate_as = 43
m.impersonate(impersonate_as)

## do something as someone else
##
##

## reset impersonation
m.clear_impersonation()
```

See also:

moco_wrapper.Moco.impersonate(), *moco_wrapper.Moco.clear_impersonation()*.

CHAPTER 5

The Moco Instance

class moco_wrapper.**Moco**(*auth={}*, *objector=<moco_wrapper.util.objector.default.DefaultObjector object>*, *requestor=<moco_wrapper.util.requestor.default.DefaultRequestor object>*, *impersonate_user_id: int = None*, ***kwargs*)

Main Moco class for handling authentication, object conversion, requesting ressources with the moco api

Parameters

- **auth** (*dict*) – Dictionary containing authentication information, see *Authentication*
- **objector** – objector object (see *Objector*, default: *moco_wrapper.util.objector.DefaultObjector*)
- **requestor** – requestor object (see *Requestor*, default: *moco_wrapper.util.requestor.DefaultRequestor*)
- **impersonate_user_id** (*int*) – user id the client should impersonate (default: None, see https://github.com/hundertzehn/mocoapp-api-docs#impersonation)

```
import moco_wrapper
moco = moco_wrapper.Moco(
    auth = {
        "api_key": "<TOKEN>",
        "domain": "<DOMAIN>"
    }
)
```

auth = None

Authentication information

It either contains an api key and and domain

```
from moco_wrapper import Moco

m = Moco(
    auth={"api_key": "here is my key", "domain": "testdomain"}
)
```

Or it contains domain, email and password

```
from moco_wrapper import Moco

m = Moco(
    auth={"domain": "testdomain", "email": "testemail@mycompany.com",
↪"password": "test"}
)
```

authenticate()
Performs any action necessary to be authenticated against the moco api.

This method gets invoked automatically, on the very first request you send against the api.

clear_impersonation()
Ends impersonation

See also:

impersonate()

delete(*path*, *params=None*, *data=None*, ***kwargs*)
Helper function for DELETE requests

full_domain
Returns the full url of the moco api

```
>> m = Moco(auth={"domain": "testabcd", ..})
>> print(m.full_domain)
https://testabcd.mocoapp.com/api/v1
```

get(*path*, *params=None*, *data=None*, ***kwargs*)
Helper function for GET requests

headers
Returns all http headers to be used by the assigned requestor

impersonate(*user_id: int*)
Impersonates the user with the supplied user id

Parameters **user_id** – user id to impersonate

See also:

clear_impersonation() to end impersonation of `user_id`

objector
Get the currently assigned objector object

See also:

Objector

patch(*path*, *params=None*, *data=None*, ***kwargs*)
Helper function for PATCH requests

post(*path*, *params=None*, *data=None*, ***kwargs*)
Helper function for POST requests

put(*path*, *params=None*, *data=None*, ***kwargs*)
Helper function for PUT requests

request (*method: str*, *path: str*, *params: dict = None*, *data: dict = None*, *bypass_auth: bool = False*, ***kwargs*)
Requests the given resource with the assigned requestor

Parameters

- **method** – HTTP Method (eg. POST, GET, PUT, DELETE)
- **path** – path of the resource (e.g. `/projects`)
- **params** – url parameters (e.g. `page=1`, query parameters)
- **data** – dictionary with data (http body)
- **bypass_auth** – If authentication checks should be skipped (default False)

The request will be given to the currently assigned requestor (see *Requestor*). The response will then be given to the currently assigned objector (see *Objector*)

The *possibly* modified response will then be returned

requestor
Get the currently assigned requestor object

See also:

Requestor

session
Get the http.session object of the current requestor (None if the requestor does not have a session)

5.1 Activity

class moco_wrapper.models.**Activity**(*moco*)
Class for handling activities.

Activities are always created for a project task. The order of things is *Project>Task>Activity*. An activity always belongs to a task and that task always belongs to a project.

Example Usage:

```
import datetime
from moco_wrapper import Moco

m = Moco()
project_id = 2
task_id = 3

#log time
created_activity = m.Activity.create(
    datetime.date(2020, 1, 1),
    project_id,
    task_id,
    0.25
    description="did things"
)
```

create(*activity_date: datetime.date*, *project_id: int*, *task_id: int*, *hours: float*, *description: str = None*, *billable: bool = None*, *tag: str = None*, *remote_service: str = None*, *remote_id: int = None*, *remote_url: str = None*)
Create an activity.

Parameters

- **activity_date** (*datetime.date, str*) – Date of the activity
- **project_id** (*int*) – Id of the project this activity belongs to
- **task_id** (*int*) – Id of the task this activity belongs to
- **hours** (*float*) – Hours to log to the activity (pass 0 to start a timer, if the date is today)
- **description** (*str*) – Activity description text (default `None`)
- **billable** (*bool*) – If this activity is billable (pass `None` if billing is dependent on project configuration) (default `None`)
- **tag** (*str*) – Tag string (default `None`)
- **remote_service** (*ActivityRemoteService*, str) – Name of the remote service referenced by the activity (default `None`)
- **remote_id** (*str*) – Id of the activity in the remote_service (default `None`)
- **remote_url** (*str*) – Url of the remote service (default `None`)

Returns The created activity

Return type *moco_wrapper.util.response.ObjectResponse*

delete (*activity_id: int*)
Delete an activity.

Parameters **activity_id** (*int*) – Id of the activity to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

disregard (*reason: str*, *activity_ids: list*, *company_id: int*, *project_id: int = None*)
Disregard activities.

Parameters

- **reason** (*str*) – Reason text for disregarding these activities
- **activity_ids** (*list*) – List of activity ids to disregard
- **company_id** (*int*) – Company id these activities belong to
- **project_id** (*int*) – Project id these activities belong to (default `None`)

Returns List with the activity ids that were disregarded

Return type *moco_wrapper.util.response.PagedListResponse*

get (*activity_id: int*)
Get a single activity.

Parameters **activity_id** (*int*) – Id of the activity

Returns The activity object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*from_date: datetime.date*, *to_date: datetime.date*, *user_id: int = None*, *project_id: int = None*, *task_id: int = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Get a list of activities.

Parameters

- **from_date** (*datetime.date, str*) – Start date

- **to_date** (*datetime.date, str*) – End date
- **user_id** (*int*) – User Id the activity belongs to (default `None`)
- **project_id** (*int*) – Id of the project the activity belongs to (default `None`)
- **task_id** (*int*) – Id of the task the activity belongs to (default `None`)
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default 1)

Returns List of activities

Return type *moco_wrapper.util.response.PagedListResponse*

start_timer(*activity_id: int*)

Start a timer on the specified activity.

Timers can only be started for activities of the current day.

Parameters **activity_id** (*int*) – Id of the activity

Returns The activity the timer was started for

Return type *moco_wrapper.util.response.ObjectResponse*

Note: Timers can only be started for activities of the current day

---

stop_timer(*activity_id: int*)

Stop a timer on the specified activity.

Parameters **activity_id** (*int*) – Id of the activity

Returns The activity the timer was stopped for

Return type *moco_wrapper.util.response.ObjectResponse*

update(*activity_id: int*, *activity_date: datetime.date = None*, *project_id: int = None*, *task_id: int = None*, *hours: float = None*, *description: str = None*, *billable: bool = None*, *tag: str = None*, *remote_service: str = None*, *remote_id: int = None*, *remote_url: str = None*)

Update an activity.

Parameters

- **activity_id** (*int*) – Id of the activity
- **activity_date** (*datetime.date, str*) – Date of the activity
- **project_id** (*int*) – Id of the project this activity belongs to
- **task_id** (*int*) – Id of the task this activity belongs to
- **hours** (*float*) – hours to log to the activity (pass 0 to start a timer, if the date is today)
- **description** (*str*) – Description text (default `None`)
- **billable** (*bool*) – If this activity is billable (pass `None`) if billing is dependent on project configuration) (default `None`)
- **tag** (*str*) – Tag string (default `None`)
- **remote_service** (*ActivityRemoteService*, str) – Name of the remote service referenced by the activity (default `None`)

- **remote_id** (*str*) – Id of the activity in the remote_service (default None)
- **remote_url** (*str*) – Url of the remote service (default None)

Returns The updated activity

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.activity.**ActivityRemoteService**

Enumeration for allowed values used that can be supplied for the remote_service argument in *Activity.create()* and *Activity.update()*

```
from moco_wrapper import Moco
from moco_wrapper.models.activity import ActivityRemoteService

m = Moco()
activity_create = m.Activity.create(
    ..
    remote_service = ActivityRemoteService.TRELLO
)
```

ASANA = 'asana'

BASECAMP = 'basecamp'

BASECAMP2 = 'basecamp2'

BASECAMP3 = 'basecamp3'

CLICKUP = 'clickup'

GITHUB = 'github'

JIRA = 'jira'

MITE = 'mite'

TOGGL = 'toggl'

TRELLO = 'trello'

WUNDERLIST = 'wunderlist'

YOUTRACK = 'youtrack'

5.2 Comment

class moco_wrapper.models.**Comment**(*moco*)

Class for handling comments.

Comments can be created for a multitude of objects. So when creating comments one must specify which type of object they target (see *CommentTargetType*)

Example Usage:

```
m = Moco()
project_id = 22
comment_create = m.Comment.create(
    project_id, #id of the thing we comment
    "PROJECT", #object type
    "example comment text"
)
```

create(*commentable_id: int*, *commentable_type: moco_wrapper.models.comment.CommentTargetType*, *text: str*)
Create a single comment.

Parameters

- **commentable_id** (*int*) – Id of the object to create the comment of (i.e the project id of the project we want to comment on)
- **commentable_type** (*CommentTargetType*, str) – Type of object to create the comment for.
- **text** (*str*) – Comment text

Returns The created comment

Return type *moco_wrapper.util.response.ObjectResponse*

create_bulk(*commentable_ids: list*, *commentable_type: moco_wrapper.models.comment.CommentTargetType*, *text: str*)
Create a comment for multiple target objects.

Parameters

- **commentable_ids** (*list*) – Ids of the objects we want to comment under ie. [123, 124, 125]
- **commentable_type** (*CommentTargetType*, str) – Type of object to create the comment for.
- **text** (*str*) – Comment text

Returns List of created comments.

Return type *moco_wrapper.util.response.ListResponse*

delete(*comment_id: int*)
Delete a comment.

Parameters **comment_id** (*int*) – Id of the comment to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get(*comment_id: int*)
Retrieve a single comment.

Parameters **comment_id** (*int*) – Id of the comment

Returns Single comment

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*commentable_type: moco_wrapper.models.comment.CommentTargetType = None*, *commentable_id: int = None*, *user_id: int = None*, *manual: bool = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of comments.

Parameters

- **commentable_type** (*CommentTargetType*, str) – Type of object the comment(s) belong to (default `None`)
- **commentable_id** (*int*) – Id of the object the comment belongs to (default `None`)
- **user_id** (*int*) – User id of the creator (default `None`)

- **manual** (*bool*) – If the comment was user-created of generated (default `None`)
- **sort_by** (*str*) – Field to sort the results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns list of comments

Return type *moco_wrapper.util.response.PagedListResponse*

update (*comment_id: int*, *text: str*)
Update a comment.

Parameters

- **comment_id** (*int*) – The id of the comment to update
- **text** (*str*) – Comment text

Returns The created comment

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.comment.**CommentTargetType**
Enumeration for allowed values used that can be supplied for the `commentable_type` argument in *Comment.create()*, *Comment.create_bulk()* and *Comment.getlist()*

```
from moco_wrapper import Moco
from moco_wrapper.models.comment import CommentTargetType

m = Moco()
comment_create = m.Comment.create(
    ..
    commentable_type = CommentTargetType.DEAL
)
```

COMPANY = 'Company'

CONTACT = 'Contact'

DEAL = 'Deal'

EXPENSE = 'Expense'

INVOICE = 'Invoice'

INVOICE_BOOKKEEPING_EXPORT = 'InvoiceBookkeepingExport'

INVOICE_DELETION = 'InvoiceDeletion'

INVOICE_REMINDER = 'InvoiceReminder'

OFFER = 'Offer'

OFFER_CONFIRMATION = 'OfferConfirmation'

PROJECT = 'Project'

PURCHASE = 'Purchase'

PURCHASE_BOOKKEEPING_EXPORT = 'PurchaseBookkeepingExport'

RECEIPT = 'Receipt'

RECEIPT_REFUND_REQUEST = 'ReceiptRefundRequest'

RECURRING_EXPENSE = 'RecurringExpense'

UNIT = 'Unit'

USER = 'User'

5.3 Company

class moco_wrapper.models.**Company**(*moco*)

Class for handling companies.

Companies come in three different flavours (see *CompanyType*), customers are companies you do stuff for and send invoices to. suppliers are companies that supply stuff to you as a customer. Finally organizations are companies that do not fit the laben customer or supplier. For the most part you will interact with companies of type customer.

Example usage:

```
from moco_wrapper import Moco

m = Moco()
new_customer = m.Company.create(
    "my new customer",
    "customer"
)
```

create(*name: str*, *company_type: moco_wrapper.models.company.CompanyType*, *website: str = None*, *fax: str = None*, *phone: str = None*, *email: str = None*, *billing_email_cc: str = None*, *address: str = None*, *info: str = None*, *custom_properties: dict = None*, *labels: list = None*, *user_id: int = None*, *currency: str = None*, *identifier: str = None*, *billing_tax: float = None*, *default_invoice_due_days: int = None*, *country_code: str = None*, *vat_identifier: str = None*, *iban: str = None*, *debit_number: int = None*, *credit_number: int = None*, *footer: str = None*)

Create a company.

Parameters

- **name** (*str*) – Name of the company
- **company_type** (*CompanyType*, str) – Either customer, supplier or organization
- **website** (*str*) – Url of the companies website (default None)
- **fax** (*str*) – Fax number of the company (default None)
- **phone** (*str*) – Phone number of the company (default None)
- **email** (*str*) – Email address of the company (default None)
- **billing_email_cc** (*str*) – Email address to cc for billing emails (default None)
- **address** (*str*) – Company address (default None)
- **info** (*str*) – Additional information about the company (default None)
- **custom_properties** (*dict*) – Custom properties dictionary (default None)
- **labels** (*list*) – Array of labels (default None)
- **user_id** (*int*) – User Id of the responsible person (default None)
- **currency** (*str*) – Currency the company uses (only customer) (default None)

- **identifier** (*str*) – Identifier of the company (only mandatory when not automatically assigned) (default None)
- **billing_tax** (*float*) – Billing tax value (from 0 to 100) (default None)
- **default_invoice_due_days** (*int*) – Payment target days for the company when creating invoices (only customer) (default None)
- **country_code** (*str*) – ISO Alpha-2 Country Code like "DE" / "CH" / "AT" in upper case - default is account country (default None)
- **vat_identifier** (*str*) – Vat identifier for eu companies (default None)
- **iban** (*str*) – Iban number (only supplier) (default None)
- **debit_number** (*int*) – Debit number (if bookkeeping is enabled) (only customer) (default None)
- **credit_number** (*int*) – Credit number (if bookkeeping is enabled) (only supplier) (default None)
- **footer** (*str*) – Some html (appears at the end of invoices) (default None)

Returns The created company

Return type *moco_wrapper.util.response.ObjectResponse*

Note: When supplying a vat_identifier, make sure it is valid

---

get (*company_id: int*)
Get a single company.

Parameters **company_id** (*int*) – Id of the company

Returns Single company object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*company_type: moco_wrapper.models.company.CompanyType = None, tags: list = None, identifier: str = None, sort_by: str = None, sort_order: str = 'asc', page: int = 1*)
Get a list of company objects.

Parameters

- **company_type** (*CompanyType*, str) – Type of company to filter for (default None)
- **tags** (*list*) – List of tags (default None)
- **identifier** (*str*) – Company identifier (default None)
- **sort_by** (*str*) – Field to sort by (default None)
- **sort_order** (*str*) – asc or desc (default "asc")
- **page** (*int*) – page number (default 1)

Returns List of companies

Return type *moco_wrapper.util.response.PagedListResponse*

update(*company_id: int*, *company_type: moco_wrapper.models.company.CompanyType = None*, *name: str = None*, *website: str = None*, *fax: str = None*, *phone: str = None*, *email: str = None*, *billing_email_cc: str = None*, *address: str = None*, *info: str = None*, *custom_properties: dict = None*, *labels: list = None*, *user_id: int = None*, *currency: str = None*, *identifier: str = None*, *billing_tax: float = None*, *default_invoice_due_days: int = None*, *country_code: str = None*, *vat_identifier: str = None*, *iban: str = None*, *debit_number: int = None*, *credit_number: int = None*, *footer: str = None*)

Update a company.

Parameters

- **company_id** (*int*) – Id of the company
- **company_type** (*CompanyType*, str) – Type of the company to modify (default `None`)
- **name** (*str*) – Name of the company (default `None`)
- **website** (*str*) – Url of the companies website (default `None`)
- **fax** (*str*) – Fax number of the company (default `None`)
- **phone** (*str*) – Phone number of the company (default `None`)
- **email** (*str*) – Email address of the company (default `None`)
- **billing_email_cc** (*str*) – Email address to cc in billing emails (default `None`)
- **address** (*str*) – Company address (default `None`)
- **info** (*str*) – Additional information about the company (default `None`)
- **custom_properties** (*dict*) – Custom properties dictionary (default `None`)
- **labels** (*list*) – Array of labels (default `None`)
- **user_id** (*int*) – Id of the responsible person (default `None`)
- **currency** (*str*) – Currency the company uses (only customer) (default `None`)
- **identifier** (*str*) – Identifier of the company (only mandatory when not automatically assigned) (only customer) (default `None`)
- **billing_tax** (*float*) – Billing tax value (only customer) (default `None`)
- **default_invoice_due_days** (*int*) – payment target days for the company when creating invoices (only customer) (default `None`)
- **country_code** (*str*) – ISO Alpha-2 Country Code like "DE" / "CH" / "AT" in upper case - default is account country (default `None`)
- **vat_identifier** (*str*) – vat identifier for eu companies (default `None`)
- **iban** (*str*) – iban number (only supplier) (default `None`)
- **debit_number** (*int*) – Debit number (if bookkeeping is enabled) (only customer) (default `None`)
- **credit_number** (*int*) – Credit number (if bookkeeping is enabled) (ony supplier) (default `None`)

Returns The updated company

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.company.**CompanyType**

Enumeration of the type of companies that exist. Can be used to supply the `company_type` argument of *Company.create()*, *Company.update()* and *Company.getlist()*

Example Usage:

```
from moco_wrapper import Moco
from moco_wrapper.models.company import CompanyType

m = Moco()
new_supplier = m.Company.create(
    ..
    company_type = CompanyType.ORGANIZATION
)
```

CUSTOMER = 'customer'

ORGANIZATION = 'organization'

SUPPLIER = 'supplier'

5.4 Contact

class moco_wrapper.models.**Contact**(*moco*)

Class for handling contacts.

create(*firstname: str*, *lastname: str*, *gender: moco_wrapper.models.contact.ContactGender*, *company_id: int = None*, *title: str = None*, *job_position: str = None*, *mobile_phone: str = None*, *work_fax: str = None*, *work_phone: str = None*, *work_email: str = None*, *work_address: str = None*, *home_address: str = None*, *home_email: str = None*, *birthday: datetime.date = None*, *info: str = None*, *tags: list = None*)

Creates a contact.

Parameters

- **firstname** (*str*) – The first name of the contact
- **lastname** (*str*) – The last name of the contact
- **gender** (*ContactGender*, str) – Gender of the contact
- **company_id** (*int*) – Id of the company the contact belongs to (default None)
- **title** (*str*) – Job title the contact has (default None)
- **job_position** (*str*) – Name of the job position this contact has (default None)
- **mobile_phone** (*str*) – Mobile phone number the contact has (default None)
- **work_fax** (*str*) – Work fax number (default None)
- **work_phone** (*str*) – Work phone number (default None)
- **work_email** (*str*) – Work email address (default None)
- **work_address** (*str*) – Physical work address (default None)
- **home_address** (*str*) – Physical home address (default None)
- **home_email** (*str*) – Home email address (default None)
- **birthday** (*datetime.date, str*) – Birthday date (default None)
- **info** (*str*) – More information about the contact (default None)
- **tags** (*list*) – Array of additional tags (default None)

Returns The created contact object

Return type *moco_wrapper.util.response.ObjectResponse*

get (*contact_id: int*)
Retrieve a single contact object

Parameters **contact_id** (*int*) – Id of the contact

Returns The contact object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*tags: list = None*, *term: str = None*, *phone: str = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of contact objects

Parameters

- **tags** (*list*) – Array of tags (default `None`)
- **term** (*str*) – Full text search (fields that are searched are name, firstname, work_email and company) (default `None`)
- **phone** (*str*) – Reverse lookup for work_phone or mobile_phone (default `None`)
- **sort_by** (*str*) – Field to the results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of contact objects

Return type *moco_wrapper.util.response.PagedListResponse*

update (*contact_id: int*, *firstname: str = None*, *lastname: str = None*, *gender: moco_wrapper.models.contact.ContactGender = None*, *company_id: int = None*, *title: str = None*, *job_position: str = None*, *mobile_phone: str = None*, *work_fax: str = None*, *work_phone: str = None*, *work_email: str = None*, *work_address: str = None*, *home_address: str = None*, *home_email: str = None*, *birthday: datetime.date = None*, *info: str = None*, *tags: list = None*)
Updates a contact.

Parameters

- **contact_id** (*int*) – Id of the contact
- **firstname** (*str*) – The first name of the contact (default `None`)
- **lastname** (*str*) – The last name of the contact (default `None`)
- **gender** (*ContactGender*, str) – Gender of the contact (default `None`)
- **company_id** (*int*) – Id of the company the contact belongs to (default `None`)
- **title** (*str*) – Job title the contact has (default `None`)
- **job_position** (*str*) – name of the job position this contact has (default `None`)
- **mobile_phone** (*str*) – Mobile phone number the contact has (default `None`)
- **work_fax** (*str*) – Work fax number (default `None`)
- **work_phone** (*str*) – Work phone number (default `None`)
- **work_email** (*str*) – Work email address (default `None`)
- **work_address** (*str*) – Physical work address (default `None`)
- **home_address** (*str*) – Physical home address (default `None`)

- **home_email** (*str*) – Home email address (default None)
- **birthday** (*datetime.date, str*) – Birthday date (default None)
- **info** (*str*) – More information about the contact (default None)
- **tags** (*list*) – Array of additional tags (default None)

Returns The updated contact object

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.contact.**ContactGender**

Enumeration for allowed values that can be supplied for the gender argument in Contact.create and Contact.update.

Example Usage:

```
from moco_wrapper.models.contact import ContactGender
from moco_wrapper import Moco

m = Moco()
new_contact = m.Contact.create(
    ..
    gender = ContactGender.MALE
)
```

FEMALE = 'F'

MALE = 'M'

UNDEFINED = 'U'

5.5 Deal

class moco_wrapper.models.**Deal**(*moco*)

Class for handling deals/leads.

create(*name: str, currency: str, money: float, reminder_date: datetime.date, user_id: int, deal_category_id: int, company_id: int = None, info: str = None, status: moco_wrapper.models.deal.DealStatus = <DealStatus.PENDING: 'pending'>*)

Create a new deal.

Parameters

- **name** (*str*) – Name of the deal
- **currency** (*str*) – Currency used (e.g. EUR, CHF)
- **money** (*float*) – How much money can be generated from this deal (e.g. 205.0)
- **reminder_date** (*datetime.date, str*) – Reminder date
- **user_id** (*int*) – Id of the user the is responsible for this lead
- **deal_category_id** (*int*) – Deal category id
- **company_id** (*int*) – Company id (default None)
- **info** (*str*) – Additional information (default None)
- **status** (*DealStatus*, str) – Current state of the deal (default *DealStatus.PENDING*)

Returns The created deal object

Return type *moco_wrapper.util.response.ObjectResponse*

get (*deal_id: int*)
Retrieve a single deal.

Parameters **deal_id** (*int*) – Id of the deal

Returns Single deal object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*status: str = None*, *tags: list = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of deal objects.

Parameters

- **status** (*DealStatus*, str) – State of deal (default `None`)
- **tags** (*list*) – Array of tags (default `None`)
- **sort_by** (*str*) – Field to order results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of deal objects

Return type *moco_wrapper.util.response.PagedListResponse*

update (*deal_id: int*, *name: str = None*, *currency: str = None*, *money: float = None*, *reminder_date: datetime.date = None*, *user_id: int = None*, *deal_category_id: int = None*, *company_id: int = None*, *info: str = None*, *status: moco_wrapper.models.deal.DealStatus = None*)
Update an existing deal.

Parameters

- **deal_id** (*int*) – Id of the deal
- **name** (*str*) – Name of the deal (default `None`)
- **currency** (*str*) – Currency used (e.g. EUR, CHF) (default `None`)
- **money** (*float*) – How much money can be generated from this deal (e.g. 205.0) (default `None`)
- **reminder_date** (*datetime.date, str*) – Reminder date (default `None`)
- **user_id** (*int*) – Id of the user that is responsible for this deal (default `None`)
- **deal_category_id** (*int*) – Deal category id (default `None`)
- **company_id** (*int*) – Company id (default `None`)
- **info** (*str*) – Additional information (default `None`)
- **status** (*DealStatus*, str) – Current state of the deal (default `None`)

Returns The updated deal object

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.deal.**DealStatus**
Enumeration for the allowed values that can be supplied for the `status` argument of *Deal.create()*, *Deal.update()* and *Deal.getlist()*.

```
from moco_wrapper.models.deal import DealStatus
from moco_wrapper import Moco

m = Moco()

deal_create = m.Deal.create(
    ..
    status = DealStatus.WON
)
```

DROPPED = 'dropped'

LOST = 'lost'

PENDING = 'pending'

POTENTIAL = 'potential'

WON = 'won'

5.6 Deal Category

class moco_wrapper.models.**DealCategory**(*moco*)

Model for handling the different deal_categories used by a pending deal.

A deal (see *moco_wrapper.models.Deal*) that is in the state PENDING (see *moco_wrapper.models.deal.DealStatus*) must be assigned to deal category. A category has a name and a probability of success (in percent).

Typically a deal that is in PENDING starts at 1% and moves into the state WON if the probability reaches 100%.

```
from moco_wrapper import Moco

m = Moco()

#create a category with 75% success probability
new_category = m.DealCategory.create(
    "Second round of negotiation",
    75
)
```

create(*name: str*, *probability: int*)

Create a new deal category.

Parameters

- **name** (*str*) – Name of the deal category (must be unique)
- **probability** (*int*) – Deal category success probability (between 1 and 100)

Returns The created deal category

Return type *moco_wrapper.util.response.ObjectResponse*

delete(*category_id: int*)

Delete a deal category.

Parameters **category_id** (*int*) – Id of the deal category to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get(*category_id: int*)
Retrieves a single deal category.

Parameters **category_id** (*int*) – Id of the deal category to retrieve

Returns Single deal category

Return type *moco_wrapper.util.response.ObjectResponse*

getlist()
Retrieves a list of a deal categories.

Returns List of deal categories

Return type *moco_wrapper.util.response.ListResponse*

update(*category_id: int*, *name: str = None*, *probability: int = None*)
Updates an existing deal category.

Parameters

- **category_id** (*int*) – Id of the deal category to update
- **name** (*str*) – Name of the deal category (must be unique) (default `None`)
- **probability** (*int*) – Deal category success probability (between 1 and 100) (default `None`)

Returns The updated deal category

Return type *moco_wrapper.util.response.ObjectResponse*

5.7 Hourly Rate

class moco_wrapper.models.**HourlyRate**(*moco*)
Model for handling hourly rates

get(*company_id: int = None*)
Get the hourly rate

Parameters **company_id** (*int*) – Company id to get the hourly rates for (default `None`)

Returns Hourly rates of the specified company

Note: When no company_id is specified the global hourly rates are returned

5.8 Invoice

class moco_wrapper.models.**Invoice**(*moco*)
Model for handling invoices.

create(*customer_id: int*, *recipient_address: str*, *created_date: datetime.date*, *due_date: datetime.date*, *service_period_from: datetime.date*, *service_period_to: datetime.date*, *title: str*, *tax: float*, *currency: str*, *items: list*, *status: moco_wrapper.models.invoice.InvoiceStatus = <InvoiceStatus.CREATED: 'created'>*, *change_address: moco_wrapper.models.invoice.InvoiceChangeAddress = <InvoiceChangeAddress.INVOICE: 'invoice'>*, *salutation: str = None*, *footer: str = None*, *discount: float = None*, *cash_discount: float = None*, *cash_discount_days: int = None*, *project_id: int = None*, *tags: list = []*)

Creates a new invoice.

Parameters

- **customer_id** (*int*) – Id of the customer/company
- **recipient_address** (*str*) – Customers address
- **created_date** (*datetime.date, str*) – Creation date of the invoice
- **due_date** (*datetime.date, str*) – Date the invoice is due
- **service_period_from** (*datetime.date, str*) – Service period start date
- **service_period_to** (*datetime.date, str*) – Service period end date
- **title** (*str*) – Title of the invoice
- **tax** (*float*) – Tax percent (between 0.0 and 100.0)
- **currency** (*str*) – Currency code (e.g. EUR)
- **items** (*list*) – Invoice items
- **status** (*InvoiceStatus*, str) – State of the invoice (default *InvoiceStatus.CREATED*)
- **change_address** (*InvoiceChangeAddress*, str) – Address propagation (default *InvoiceChangeAddress.INVOICE*)
- **salutation** (*str*) – Salutation text (default `None`)
- **footer** (*str*) – Footer text (default `None`)
- **discount** (*float*) – Discount in percent (between 0.0 and 100.0) (default `None`)
- **cash_discount** (*float*) – Cash discount in percent (between 0.0 and 100.0) (default `None`)
- **cash_discount_days** (*float*) – How many days is the cash discount valid (ex. 4) (default `None`)
- **project_id** (*int*) – Id of the project the invoice belongs to (default `None`)
- **tags** (*list*) – List of tags (default `[]`)

Returns The created invoice

Return type *moco_wrapper.util.response.ObjectResponse*

Note: Note that if you create an invoice with a project, that project must also belong to the customer the invoice was created for.

See also:

moco_wrapper.util.generator.InvoiceItemGenerator

get(*invoice_id: int*)
Retrieve a single invoice.

Parameters **invoice_id** (*int*) – Invoice id

Returns Single invoice object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*status: moco_wrapper.models.invoice.InvoiceStatus = None*, *date_from: datetime.date = None*, *date_to: datetime.date = None*, *tags: list = None*, *identifier: str = None*, *term: str = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of invoices.

Parameters

- **status** (*InvoiceStatus*, str) – State of the invoice (default `None`)
- **date_from** (*datetime.date, str*) – Starting date (default `None`)
- **date_to** (*datetime.date, str*) – End date (default `None`)
- **tags** (*list*) – List of tags (default `None`)
- **identifier** (*str*) – Identifier string (e.g. R1903-003) (default `None`)
- **term** (*str*) – Wildcard search term (default `None`)
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of invoice objects

Return type *moco_wrapper.util.response.PagedListResponse*

locked(*status: moco_wrapper.models.invoice.InvoiceStatus = None*, *date_from: datetime.date = None*, *date_to: datetime.date = None*, *identifier: str = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of locked invoices.

Parameters

- **status** (*InvoiceStatus*, str) – State of the invoice (default `None`)
- **date_from** (*datetime.date, str*) – Start date (default `None`)
- **date_to** (*datetime.date, str*) – End date (default `None`)
- **identifier** (*str*) – Identifier string (ex. R1903-003) (default `None`)
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of invoice objects

Return type *moco_wrapper.util.response.PagedListResponse*

pdf(*invoice_id: int*)
Retrieve the invoice document as pdf.

Parameters **invoice_id** (*int*) – Invoice id

Returns Invoice pdf

Return type *moco_wrapper.util.response.FileResponse*

send_email(*invoice_id: int*, *emails_to: str*, *subject: str*, *text: str*, *emails_cc: str = None*, *emails_bcc: str = None*)
Send an invoice by mail

Parameters

- **invoice_id** (*int*) – Id of the invoice to send
- **emails_to** (*str, list*) – Target email address (or a list of multiple email addresses)
- **subject** (*str*) – Email subject
- **text** (*str*) – Email text
- **emails_cc** (*str, list*) – Email address for cc (or a list of multiple email addresses) (default `None`)
- **emails_bcc** (*str, list*) – Email address for bcc (or a list of multiple email addresses) (default `None`)

Returns Object containing the details of the sent mail

Return type *moco_wrapper.util.response.ObjectResponse*

Note: If you want to send an email to the default recipient configured in the project or customer, set `emails_to` and `emails_cc` To `None`.

---

timesheet(*invoice_id: int*)
Retrieve the invoice timesheet document as pdf.

Note: Invoices that have timesheets cannot be created over the api and must be created manually by billing unbilled tasks.

Parameters **invoice_id** (*int*) – Invoice id

Returns Invoice timesheet as pdf

Return type *moco_wrapper.util.response.FileResponse*

update_status(*invoice_id: int*, *status: moco_wrapper.models.invoice.InvoiceStatus*)
Updates the state of an invoices.

Parameters

- **invoice_id** (*int*) – Invoice id
- **status** (*InvoiceStatus*, str) – New state of the invoice

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

class moco_wrapper.models.invoice.**InvoiceStatus**
Enumeration for allowed values that can be supplied for the `status` argument of *Invoice.getlist()*, *Invoice.update_status()* and *Invoice.create()*.

Example usage:

```
from moco_wrapper.models.invoice import InvoiceStatus
from moco_wrapper import Moco

m = Moco()
new_invoice = m.Invoice.create(
    ..
    status = InvoiceStatus.DRAFT
)
```

CREATED = 'created'

DRAFT = 'draft'

IGNORED = 'ignored'

> **Warning:** Do not use `IGNORED` for creating invoices, only updating and filtering.

OVERDUE = 'overdue'

PAID = 'paid'

PARTIALLY_PAID = 'partially_paid'

SENT = 'sent'

class moco_wrapper.models.invoice.**InvoiceChangeAddress**

Enumeration for allowed values that can be supplied for `change_address` argument of *Invoice.create()*.

```
from moco_wrapper.models.invoice import InvoiceChangeAddress
from moco_wrapper import Moco

m = Moco()
new_invoice = m.Invoice.create(
    ..
    change_address = InvoiceChangeAddress.PROJECT
)
```

CUSTOMER = 'customer'

INVOICE = 'invoice'

PROJECT = 'project'

5.9 Invoice Payment

class moco_wrapper.models.**InvoicePayment**(*moco*)

Class for handling invoice payments.

create(*payment_date: datetime.date*, *invoice_id: int*, *paid_total: float*, *currency: str*)

Create a new invoice payment.

Parameters

- **payment_date** (*datetime.date, str*) – Date of the payment
- **invoice_id** (*int*) – Id of the invoice this payment belongs to

- **paid_total** (*float*) – Amount that was paid
- **currency** (*str*) – Currency used (e.g. EUR)

Returns The created invoice payment object

Return type *moco_wrapper.util.response.ObjectResponse*

create_bulk(*items: list = []*)

Create multiple new invoice payments.

Parameters **items** (*list*) – Payment items

Returns List of created invoice payments

Return type *moco_wrapper.util.response.ListResponse*

Bulk creation if invoice payments items with generator:

```
from moco_wrapper.util.generator import InvoicePaymentGenerator()
from moco_wrapper import Moco

items = [
    gen.generate(..),
    gen.generate(..)
]

m = Moco()

created_payments = m.InvoicePayment.create_bulk(items)
```

See also:

moco_wrapper.util.generator.InvoicePaymentGenerator

delete(*payment_id: int*)

Deletes an invoice payment.

Parameters **payment_id** (*int*) – Id of the payment to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get(*payment_id: int*)

Retrieve a single invoice payment.

Parameters **payment_id** (*int*) – Invoice payment id

Returns Single invoice payment object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*invoice_id: int = None*, *date_from: datetime.date = None*, *date_to: datetime.date = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)

Retrieve a list of invoice payments.

Parameters

- **invoice_id** (*int*) – Id of a corresponding invoice (default `None`)
- **date_from** (*datetime.date, str*) – Start date (default `None`)
- **date_to** (*datetime.date, str*) – End date (default `None`)
- **sort_by** (*str*) – Field to sort results by (default `None`)

- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of invoice payments

Return type *moco_wrapper.util.response.PagedListResponse*

update(*payment_id: int*, *payment_date: datetime.date = None*, *paid_total: float = None*, *currency: str = None*)

Updates an existing invoice payment.

Parameters

- **payment_id** (*int*) – Id of the payment to update
- **payment_date** (*datetime.date, str*) – Date of the payment (default `None`)
- **paid_total** (*float*) – Amount that was paid (default `None`)
- **currency** (*str*) – Currency (e.g. EUR) (default `None`)

Returns The updated invoice payment object

Return type *moco_wrapper.util.response.ObjectResponse*

5.10 Offer

class moco_wrapper.models.**Offer**(*moco*)

Class for handling offers.

create(*deal_id: int*, *project_id: int*, *recipient_address: str*, *creation_date: datetime.date*, *due_date: datetime.date*, *title: str*, *tax: float*, *currency: str*, *items: list*, *change_address: moco_wrapper.models.offer.OfferChangeAddress = <OfferChangeAddress.OFFER: 'offer'>*, *salutation: str = None*, *footer: str = None*, *discount: float = None*, *contact_id: int = None*)

Create a new offer.

Parameters

- **deal_id** (*int*) – Deal id of the offer
- **project_id** (*int*) – project id of the offer
- **recipient_address** (*str*) – Address of the recipient
- **creation_date** (*datetime.date, str*) – Creation date
- **due_date** (*datetime.date, str*) – Date the offer is due
- **title** (*str*) – Title of the offer
- **tax** (*float*) – Tax (0.0-100.0)
- **currency** (*str*) – Currency code used (e.g. EUR, CHF)
- **items** (*list*) – List of offer items
- **change_address** (*OfferChangeAddress*, str) – change offer address propagation (default *OfferChangeAddress.OFFER*)
- **salutation** (*str*) – Salutation text (default `None`)
- **footer** (*str*) – Footer text (default `None`)
- **discount** (*float*) – Discount in percent (default `None`)

- **contact_id** (*int*) – Id of the contact for the offer (default None)

Returns The created offer

Return type *moco_wrapper.util.response.ObjectResponse*

Note: Either deal_id or project_id must be specified (or both)

See also:

moco_wrapper.util.generator.OfferItemGenerator

get (*offer_id: int*)
Retrieve a single offer.

Parameters **offer_id** (*int*) – Id of the offer

Returns Single offer object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*status: moco_wrapper.models.offer.OfferStatus = None*, *from_date: datetime.date = None*, *to_date: datetime.date = None*, *identifier: str = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of offers.

Parameters

- **status** (*OfferStatus*, str) – State the offer is in (default None)
- **from_date** (*datetime.date, str*) – Start date (default None)
- **to_date** (*datetime.date, str*) – End date (default None)
- **identifier** (*str*) – Identifier string (e.g.: "A1903-003") (default None)
- **sort_by** (*str*) – Field to sort the results by (default None)
- **sort_order** (*str*) – asc or desc (default "asc")
- **page** (*int*) – Page number (default 1)

Returns List of offer objects

Return type *moco_wrapper.util.response.PagedListResponse*

pdf (*offer_id: int*)
Retrieve the offer document for a single offer.

Parameters **offer_id** (*int*) – Id of the offer

Returns The offers pdf document

Return type *moco_wrapper.util.response.FileResponse*

update_status (*offer_id: int*, *status: moco_wrapper.models.offer.OfferStatus*)
Updates the state of an offer

Parameters

- **offer_id** (*int*) – Id of the offer
- **status** (*OfferStatus*, str) – The new state for the offer

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

class moco_wrapper.models.offer.**OfferStatus**
Enumeration for allowed values of the `status` argument of *Offer.getlist()*.

Example usage:

```
from moco_wrapper.models.offer import OfferStatus
from moco_wrapper import Moco

m = Moco()
sent_offers = m.Offer.getlist(
    ..
    status = OfferStatus.SENT
)
```

ACCEPTED = 'accepted'

ARCHIVED = 'archived'

BILLED = 'billed'

CREATED = 'created'

PARTIALLY_BILLED = 'partially_billed'

SENT = 'sent'

class moco_wrapper.models.offer.**OfferChangeAddress**
Enumeration for allowed values of the `change_address` argument of *Offer.create()*.

Example usage:

```
from moco_wrapper.models.offer import OfferChangeAddress
from moco_wrapper import Moco

m = Moco()
new_offer = m.Offer.create(
    ..
    change_address = OfferChangeAddress.CUSTOMER
)
```

CUSTOMER = 'customer'

OFFER = 'offer'

5.11 Project

class moco_wrapper.models.**Project**(*moco*)
Class for handling projects.

archive(*project_id: int*)
Archive a project.

Parameters **project_id** (*int*) – Id of the project to archive

Returns The archived project

Return type *moco_wrapper.util.response.ObjectResponse*

assigned(*active: bool = None*)
Get list of all project currently assigned to the user.

Parameters **active** (*bool*) – Show only active or inactive projects (default None)

Returns List of project objects

Return type *moco_wrapper.util.response.ListResponse*

create(*name: str*, *currency: str*, *leader_id: int*, *customer_id: int*, *deal_id: int = None*, *finish_date: datetime.date = None*, *identifier: str = None*, *billing_address: str = None*, *billing_email_to: str = None*, *billing_email_cc: str = None*, *billing_notes: str = None*, *setting_include_time_report: bool = None*, *billing_variant: moco_wrapper.models.project.ProjectBillingVariant = None*, *hourly_rate: float = None*, *budget: float = None*, *labels: list = None*, *custom_properties: dict = None*, *info: str = None*, *fixed_price: bool = False*)

Create a new project.

Parameters

- **name** (*str*) – Name of the project
- **currency** (*str*) – Currency used by the project (e.g. EUR)
- **leader_id** (*int*) – User id of the project leader
- **customer_id** (*int*) – Company id of the customer
- **deal_id** (*int*) – Deal id the the project originated from
- **finish_date** (*datetime.date, str*) – Finish date (default None)
- **identifier** (*str*) – Project Identifier (default None)
- **billing_address** (*str*) – Billing address the invoices go to (default None)
- **billing_email_to** (*str*) – Email address to send billing email to (default None)
- **billing_email_cc** (*str*) – Email address to cc in billing emails (default None)
- **billing_notes** (*str*) – Billing notes (default None)
- **setting_include_time_report** (*bool*) – Include time report in billing emails (default None)
- **billing_variant** (*ProjectBillingVariant*, str) – Billing variant used (default None)
- **hourly_rate** (*float*) – Hourly rate that will get billed (default None)
- **budget** (*float*) – Budget for the project (default None)
- **labels** (*list*) – Array of additional labels (default None)
- **custom_properties** (*dict*) – Custom values used by the project (default None)
- **info** (*str*) – Additional information (default None)
- **fixed_price** (*bool*) – If the project is a fixed price projects (default False)

Returns The created project object

Return type *moco_wrapper.util.response.ObjectResponse*

Note: The parameter identifier is required if number ranges are manual.

Note: If you create a project with fixed_price = True, budget also has to be specified

---

get (*project_id: int*)
Get a single project.

Parameters **project_id** (*int*) – Id of the project

Returns Project object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*include_archived: bool = None*, *include_company: bool = None*, *leader_id: int = None*, *company_id: int = None*, *created_from: datetime.date = None*, *created_to: datetime.date = None*, *updated_from: datetime.date = None*, *updated_to: datetime.date = None*, *tags: list = None*, *identifier: str = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Get a list of projects.

Parameters

- **include_archived** (*bool*) – Include archived projects (default `None`)
- **include_company** (*bool*) – Include the complete company object or just the company id (default `None`)
- **leader_id** (*int*) – User id of the project leader (default `None`)
- **company_id** (*int*) – Company id the projects are assigned to (default `None`)
- **created_from** (*datetime.date, str*) – Created start date (default `None`)
- **created_to** (*datetime.date, str*) – Created end date (default `None`)
- **updated_from** (*datetime.date, str*) – Updated start date (default `None`)
- **updated_to** (*datetime.date, str*) – Updated end date (default `None`)
- **tags** (*list*) – Array of tags (default `None`)
- **identifier** – Project identifier (default `None`)
- **sort_by** (*str*) – Field to sort the results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of project objects

Return type *moco_wrapper.util.response.PagedListResponse*

report (*project_id: int*)
Retrieve a project report.

Parameters **project_id** (*int*) – Id of the project

Returns Report with the most important project business indicators

Return type *moco_wrapper.util.response.ObjectResponse*

Note: All costs are in the accounts main currency, it might differ from the budget and billable items.

---

unarchive (*project_id: int*)
Unarchive a project.

Parameters **project_id** (*int*) – Id of the project to unarchive

Returns The unarchived project

Return type *moco_wrapper.util.response.ObjectResponse*

update(*project_id: int*, *name: str = None*, *leader_id: int = None*, *customer_id: int = None*, *deal_id: int = None*, *finish_date: datetime.date = None*, *identifier: str = None*, *billing_address: str = None*, *billing_email_to: str = None*, *billing_email_cc: str = None*, *billing_notes: str = None*, *setting_include_time_report: bool = None*, *billing_variant: moco_wrapper.models.project.ProjectBillingVariant = None*, *hourly_rate: float = None*, *budget: float = None*, *labels: list = None*, *custom_properties: dict = None*, *info: str = None*)

Update an existing project.

Parameters

- **project_id** (*int*) – Id of the project to update
- **name** (*str*) – Name of the project (default None)
- **leader_id** (*int*) – User id of the project leader (default None)
- **customer_id** (*int*) – Company id of the customer (default None)
- **deal_id** (*int*) – Deal id of the project (default None)
- **finish_date** (*datetime.date, str*) – Finish date (default None)
- **identifier** (*str*) – Project Identifier (default None)
- **billing_address** (*str*) – Address the invoices go to (default None)
- **billing_email_to** (*str*) – Email address to send billing emails to (default None)
- **billing_email_cc** (*str*) – Email address to cc in billing emails (default None)
- **billing_notes** (*str*) – Billing notes
- **setting_include_time_report** (*bool*) – Include time reports in billing emails
- **billing_variant** (*ProjectBillingVariant*, str) – Billing variant used (default None)
- **hourly_rate** (*float*) – Hourly rate that will get billed (default None)
- **budget** (*float*) – Budget for the project (default None)
- **labels** (*list*) – Array of additional labels (default None)
- **custom_properties** (*dict*) – Custom values used by the project (default None)
- **info** (*str*) – Additional information (default None)

Returns The updated project object

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.project.**ProjectBillingVariant**

Enumeration for allowed values of the `billing_variant` argument of *Project.create()* and *Project.update()*.

Example usage:

```
from moco_wrapper.models.project import ProjectBillingVariant
from moco_wrapper import Moco

m = Moco()
new_project = m.Project.create(
    ..
    billing_variant = ProjectBillingVariant.USER
)
```

```
PROJECT = 'project'

TASK = 'task'

USER = 'user'
```

5.12 Project Contract

class moco_wrapper.models.**ProjectContract**(*moco*)

Class for handling project contracts.

When a user gets assigned to a project, that is called a project contract. This can be done with this model.

create(*project_id: int*, *user_id: int*, *billable: bool = None*, *active: bool = None*, *budget: float = None*, *hourly_rate: float = None*)

Assign a user to a project.

Parameters

- **project_id** (*int*) – Id of the project
- **user_id** (*int*) – User id of the person to assign
- **billable** (*bool*) – If the contract is billable (default `None`)
- **active** (*bool*) – If the contract is active (default `None`)
- **budget** (*float*) – Contract budget (default `None`)
- **hourly_rate** (*float*) – Contract hourly rate (default `None`)

Returns Created contract object

Return type *moco_wrapper.util.response.ObjectResponse*

delete(*project_id: int*, *contract_id: int*)

Delete a project contract.

Deleting a staff assignment is only possible as long as there no hours tracked from the assigned person for the project.

Parameters

- **project_id** (*int*) – Id of the project
- **contract_id** (*int*) – Id of the contract to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get(*project_id: int*, *contract_id: int*)

Retrieve a project contract.

Parameters

- **project_id** (*int*) – Id of the project
- **contract_id** (*int*) – Id of the contract

Returns The contract object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*project_id: int*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)

Retrieve all active contracts for a project.

Parameters

- **project_id** (*int*) – Id of the project
- **sort_by** (*str*) – Sort by field (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of contract objects

Return type *moco_wrapper.util.response.PagedListResponse*

update(*project_id: int*, *contract_id: int*, *billable: bool = None*, *active: bool = None*, *budget: float = None*, *hourly_rate: float = None*)

Update an existing project contract.

Parameters

- **project_id** (*int*) – Id of the project to update the contract for
- **contract_id** (*int*) – Id of the contract to update
- **billable** (*bool*) – If the contract is billable (default `None`)
- **active** (*bool*) – If the contract is active (default `None`)
- **budget** (*float*) – Contract budget (default `None`)
- **hourly_rate** (*float*) – Contract hourly rate (default `None`)

Returns The updated project contract

Return type *moco_wrapper.util.response.ObjectResponse*

5.13 Project Expense

class moco_wrapper.models.**ProjectExpense**(*moco*)

Class for handling additional project expenses.

An example for this would be when a third part product (one time cost) is bought for a specific customer project and then get billed to the project to regain the cost.

See also:

moco_wrapper.models.ProjectRecurringExpense for repeating expenses.

create(*project_id: int*, *expense_date: datetime.date*, *title: str*, *quantity: float*, *unit: str*, *unit_price: float*, *unit_cost: float*, *description: str = None*, *billable: bool = True*, *budget_relevant: bool = False*, *custom_properties: dict = None*)

Create a project expense.

Parameters

- **project_id** (*int*) – Id of the project to create the expense for
- **expense_date** (*datetime.date, str*) – Date of the expense
- **title** (*str*) – Expense title
- **quantity** (*float*) – Quantity (how much of `unit` was bought?)
- **unit** (*str*) – Name of the unit (What was bought for the customer/project?)
- **unit_price** (*float*) – Price of the unit that will be billed to the customer/project

- **unit_cost** (*float*) – Cost that we had to pay
- **description** (*str*) – Description of the expense (default `None`)
- **billable** (*bool*) – If this expense billable (default `True`)
- **budget_relevant** (*bool*) – If this expense is budget relevant (default `False`)
- **custom_properties** (*dict*) – Additional fields as dictionary (default `None`)

Returns The created expense object

Return type *moco_wrapper.util.response.ObjectResponse*

create_bulk(*project_id: int*, *items: list*)
Create multiple expenses for a project.

Parameters

- **project_id** (*int*) – Id of the project to created the expenses for
- **items** (*list*) – Items to create bulk

Returns The created entries

Return type *moco_wrapper.util.response.ListResponse*

See also:

moco_wrapper.util.generator.ProjectExpenseGenerator

delete(*project_id: int*, *expense_id: int*)
Deletes an expense.

Parameters

- **project_id** (*int*) – Id of the project the expense belongs to
- **expense_id** (*int*) – Id of the expense to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

disregard(*project_id: int*, *expense_ids: list*, *reason: str*)
Disregard expenses

Parameters

- **project_id** (*int*) – Id of the project
- **expense_ids** (*list*) – Array of expense ids to disregard
- **reason** (*str*) – Reason for disregarding the expenses

Returns List of disregarded expense ids

Return type *moco_wrapper.util.response.ListResponse*

Example usage:

```
from moco_wrapper import Moco

m = Moco()

m.ProjectExpense.disregard(
    project_id=22,
    expense_ids=[444, 522, 893],
```

(continues on next page)

(continued from previous page)

```
    reason="Expenses already billed"
)
```

get (*project_id: int*, *expense_id: int*)
Retrieve a single expense object.

Parameters

- **project_id** (*int*) – Id of the project
- **expense_id** (*int*) – If of the expense to retrieve

Returns Single expense object

Return type *moco_wrapper.util.response.ObjectResponse*

getall (*from_date: datetime.date = None*, *to_date: datetime.date = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Get a list of all expenses.

Parameters

- **from_date** (*datetime.date, str*) – Start date (default `None`)
- **to_date** (*datetime.date, str*) – End date (default `None`)
- **sort_by** (*str*) – Sort results by field (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of expense objects

Return type *moco_wrapper.util.response.PagedListResponse*

getlist (*project_id: int*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve all expenses of a project.

Parameters

- **project_id** (*int*) – Id of the project
- **sort_by** (*str*) – Sort results by field (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of expense objects

Return type *moco_wrapper.util.response.PagedListResponse*

update (*project_id: int*, *expense_id: int*, *expense_date: datetime.date = None*, *title: str = None*, *quantity: float = None*, *unit: str = None*, *unit_price: float = None*, *unit_cost: float = None*, *description: str = None*, *billable: bool = None*, *budget_relevant: bool = None*, *custom_properties: dict = None*)
Update an existing project expense.

Parameters

- **project_id** (*int*) – Id of the project
- **expense_id** (*int*) – id of the expense we want to update
- **expense_date** (*datetime.date, str*) – Date of the expense (default `None`)

- **title** (*str*) – Expense title (default None)
- **quantity** (*float*) – Quantity (how much of unit was bought?) (default None)
- **unit** (*str*) – Name of the unit (What was bought for the customer/project?) (default None)
- **unit_price** (*float*) – Price of the unit that will be billed to the customer/project (default None)
- **unit_cost** (*float*) – Cost that we had to pay (default None)
- **description** (*str*) – Description of the expense (default None)
- **billable** (*bool*) – If this expense billable (default None)
- **budget_relevant** (*bool*) – If this expense is budget relevant (default None)
- **custom_properties** (*dict*) – Additional fields as dictionary (default None)

Returns The updated expense object

Return type *moco_wrapper.util.response.ObjectResponse*

5.14 Project Recurring Expense

class moco_wrapper.models.**ProjectRecurringExpense**(*moco*)

Class for handling recurring expenses of a project.

An example for this would be when a third part subscription (repeat cost) is bought for a specific customer project and then get billed to the project to regain the cost.

See also:

moco_wrapper.models.ProjectExpense for one time expenses.

create(*project_id: int*, *start_date: datetime.date*, *period: moco_wrapper.models.project_recurring_expense.ProjectRecurringE*
title: str, *quantity: float*, *unit: str*, *unit_price: float*, *unit_cost: float*, *finish_date: datetime.date = None*, *description: str = None*, *billable: bool = True*, *budget_relevant: bool = False*, *custom_properties: dict = None*)

Create a new recurring expense for a project.

Parameters

- **project_id** (*int*) – Id of the project to create the expense for
- **start_date** (*datetime.date, str*) – Starting date of the expense
- **period** (*ProjectRecurringExpensePeriod*, str) – period of the expense
- **title** (*str*) – Title of the expense
- **quantity** (*float*) – Quantity (how much of unit was bought?)
- **unit** (*str*) – Name of the unit (What was bought for the customer/project?)
- **unit_price** (*float*) – Price of the unit that will be billed to the customer/project
- **unit_cost** (*float*) – Cost that we had to pay
- **finish_date** (*datetime.date, str*) – Finish date, (if None: unlimited) (default None)
- **description** (*str*) – Description of the expense (default None)

- **billable** (*bool*) – If this expense billable (default `True`)
- **budget_relevant** (*bool*) – If this expense is budget relevant (default `False`)
- **custom_properties** (*dict*) – Additional fields as dictionary (default `None`)

Returns The created recurring expense object

Return type *moco_wrapper.util.response.ObjectResponse*

delete (*project_id: int*, *recurring_expense_id: int*)
Deletes an existing recurring expense.

Parameters

- **project_id** (*int*) – Project id the expense belongs to
- **recurring_expense_id** (*int*) – Id of the expense to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get (*project_id: int*, *recurring_expense_id: int*)
Retrieve a single recurring expense

Parameters

- **project_id** (*int*) – Id of the project the expense belongs to
- **recurring_expense_id** (*int*) – iI of the recurring expense

Returns Single recurring expense object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*project_id: int*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of recurring expenses for a project.

Parameters

- **project_id** (*int*) – Id of the project the expenses belong to
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of recurring expenses

Return type *moco_wrapper.util.response.PagedListResponse*

update (*project_id: int*, *recurring_expense_id: int*, *title: str = None*, *quantity: float = None*, *unit: str = None*, *unit_price: float = None*, *unit_cost: float = None*, *finish_date: datetime.date = None*, *description: str = None*, *billable: bool = None*, *budget_relevant: bool = None*, *custom_properties: dict = None*)
Update an existing recurring expense.

Parameters

- **project_id** (*int*) – Id of the project
- **recurring_expense_id** (*int*) – Id of the recurring expense to update
- **title** (*str*) – Title of the expense (default `None`)
- **quantity** (*int*) – Quantity (how much of `unit` was bought?) (default `None`)

- **unit** (*str*) – Name of the unit (What was bought for the customer/project?) (default None)
- **unit_price** (*float*) – Price of the unit that will be billed to the customer/project (default None)
- **unit_cost** (*float*) – Cost that we had to pay (default None)
- **finish_date** (*datetime.date, str*) – Finish date, (if None: unlimited) (default None)
- **description** (*str*) – Description of the expense (default None)
- **billable** (*bool*) – If this expense billable (default None)
- **budget_relevant** – If this expense is budget relevant (default None)
- **custom_properties** (*dict*) – Additional fields as dictionary (default None)

Returns The updated recurring expense object

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.project_recurring_expense.**ProjectRecurringExpensePeriod**

Enumeration for allowed values of `period` argument of *ProjectRecurringExpense.create()*.

Example usage:

```
from moco_wrapper.models.project_recurring_expense import␣
↪ProjectRecurringExpensePeriod
from moco_wrapper import Moco

m = Moco()
recur_expense = m.ProjectRecurringExpense.create(
    ..
    period = ProjectRecurringExpensePeriod.WEEKLY
)
```

ANNUAL = 'annual'

BIANNUAL = 'biannual'

BIWEEKLY = 'biweekly'

MONTHLY = 'monthly'

QUARTERLY = 'quarterly'

WEEKLY = 'weekly'

5.15 Project Payment Schedule

class moco_wrapper.models.**ProjectPaymentSchedule**(*moco*)

Class for handling billing schedules for fixed price projects.

Fixed Price projects can have a target date they should be billed on (in the future). With this class you can create this target entry (and how much should be billed).

For Example you can create a project that will be billed in four (4) steps over the year.

```
from moco_wrapper import Moco
from datetime import date

m = Moco()

# create fixed price project
project = m.Project.create(
    name="my fixed price project",
    currency="EUR",
    leader_id=1,
    customer_id=2,
    fixed_price=True,
    budget=4000
).data

first_payment = m.ProjectPaymentSchedule.create(project.id, 1000.0, date(2020, 3,␣
↪1))
second_payment = m.ProjectPaymentSchedule.create(project.id, 1000.0, date(2020, 6,
↪ 1))
third_payment = m.ProjectPaymentSchedule.create(project.id, 1000.0, date(2020, 9,␣
↪1))
fourth_payment = m.ProjectPaymentSchedule.create(project.id, 1000.0, date(2020,␣
↪12, 1))
```

See also:

moco_wrapper.models.Project.create()

create (*project_id: int*, *net_total: float*, *schedule_date: datetime.date*, *title: str = None*, *checked: bool = False*)

Creates a new project payment schedule.

Parameters

- **project_id** (*int*) – The id of the project the entry belongs to
- **net_total** (*float*) – How much should be billed on this schedule
- **schedule_date** (*datetime.date, str*) – Date of the entry
- **title** (*str*) – Title string (default `None`)
- **checked** (*bool*) – Mark entry as checked (the entry will be crossed out in the UI) (default `False`)

Returns The created schedule item

Return type *moco_wrapper.util.response.ObjectResponse*

delete (*project_id: int*, *schedule_id: int*)

Delete a project payment schedule item

Parameters

- **project_id** (*int*) – Project the payment schedule belongs to
- **schedule_id** (*int*) – Id of the schedule item to delete

Returns The deleted response on success

Return type *moco_wrapper.util.response.ObjectResponse*

get (*project_id: int*, *schedule_id: int*)

Retrieves project payment schedule.

Parameters

- **project_id** (*int*) – Id of the project to schedule item belongs to
- **schedule_id** (*int*) – Id of the schedule to retrieve

Returns The schedule item

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*project_id: int*)
Retrieve a list of project payment schedules

Parameters **project_id** (*int*) – Project id of the schedule items

Returns List of schedules payments

Return type *moco_wrapper.util.response.ListResponse*

update (*project_id: int*, *schedule_id: int*, *net_total: float = None*, *schedule_date: datetime.date = None*, *title: str = None*, *checked: bool = None*)
Updates an existing project payment schedule.

Parameters

- **project_id** (*int*) – Project id the schedule item belongs to
- **schedule_id** (*int*) – Id of the schedule item to update
- **net_total** (*float*) – Total amount to be billed (default `None`)
- **schedule_date** (*datetime.date, str*) – Date the billing will take place (default `None`)
- **title** (*str*) – Title of the item (default `None`)
- **checked** (*bool*) – Mark entry as checked (the entry will be crossed out in the UI) (default `None`)

Returns The updated schedule item

Return type *moco_wrapper.util.response.ObjectResponse*

5.16 Project Task

class moco_wrapper.models.**ProjectTask** (*moco*)
Class for handling tasks of a project.

create (*project_id: int*, *name: str*, *billable: bool = True*, *active: bool = True*, *budget: float = None*, *hourly_rate: float = None*)
Create a task for a project.

Parameters

- **project_id** (*int*) – Id of the project the created task will belong to
- **name** (*str*) – Name of the task
- **billable** (*bool*) – If this expense billable (default `True`)
- **active** (*bool*) – If this expense active (default `True`)
- **budget** (*float*) – Budget for the task (e.g. 200.75) (default `None`)
- **hourly_rate** (*float*) – How much is the hourly rate for the task (e.g.: 120.5) (default `None`)

Returns The created project task

Return type *moco_wrapper.util.response.ObjectResponse*

delete(*project_id: int*, *task_id: int*)
Delete project task.

Parameters

- **project_id** (*int*) – Id of the project the task belongs to
- **task_id** (*int*) – Id of the task to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

Note: Deletion of a task is only possible as long as no hours were tracked for the task

---

get(*project_id: int*, *task_id: int*)
Retrieve a single project task.

Parameters

- **project_id** (*int*) – Id of the project the task belongs to
- **task_id** (*int*) – Id of the task

Returns Single project task

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*project_id: int*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of tasks for a project.

Parameters

- **project_id** (*int*) – Id of the project
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of project tasks

Return type *moco_wrapper.util.response.PagedListResponse*

update(*project_id: int*, *task_id: int*, *name: str = None*, *billable: bool = None*, *active: bool = None*, *budget: float = None*, *hourly_rate: float = None*)
Update a task for a project.

Parameters

- **project_id** (*int*) – Id of the project the task belongs to
- **task_id** (*int*) – Id of the task to update
- **name** (*str*) – Name of the task (default `None`)
- **billable** (*bool*) – If this expense billable (default `None`)
- **active** (*bool*) – If this expense active (default `None`)
- **budget** (*float*) – Budget for the task (e.g. 200.75) (default `None`)

- **hourly_rate** (*float*) – How much is the hourly rate for the task (e.g.: 120.5) (default None)

Returns The updated project task

Return type *moco_wrapper.util.response.ObjectResponse*

5.17 Planning Entry

class moco_wrapper.models.**PlanningEntry**(*moco*)
Class for handling planning.

Note: This is the new way for handling planning (the old way was with Schedules)

---

create(*project_id: int, starts_on: datetime.date, ends_on: datetime.date, hours_per_day: float, user_id: int = None, comment: str = None, symbol: moco_wrapper.models.planning_entry.PlanningEntrySymbol = None*)
Create a new planning entry.

Parameters

- **project_id** (*int*) – Project id
- **starts_on** (*datetime.date, str*) – Start date
- **ends_on** (*datetime.date, str*) – End date
- **hours_per_day** (*float*) – Hours per day the planning entry will consume
- **user_id** (*int*) – User id the planning entry belongs to (default None)
- **comment** (*str*) – A comment (default None)
- **symbol** (*PlanningEntrySymbol*, int) – Symbol icon to use (default None)

Returns The created planning entry

Return type *moco_wrapper.util.response.ObjectResponse*

Note: If no user_id is supplied the entry will be created with the user_id of the executing request (the user_id the api key belongs to)

---

delete(*planning_entry_id: int*)
Delete a planning entry.

Parameters **planning_entry_id** (*int*) – Id of the entry to delete

Returns The deleted planning entry

Return type *moco_wrapper.util.response.ObjectResponse*

get(*planning_entry_id: int*)
Retrieve a single planning entry.

Parameters **planning_entry_id** (*int*) – Id the of the entry

Returns Single planning entry

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*start_date: datetime.date = None*, *end_date: datetime.date = None*, *user_id: int = None*, *project_id: int = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of planning entries.

Parameters

- **start_date** (*datetime.date, str*) – Start date (if `start_date` is supplied, *end_date* must also be supplied) (default `None`)
- **end_date** (*datetime.date, str*) – End date (if *end_date* is supplied, `start_date` must also be supplied) (default `None`)
- **user_id** (*int*) – User id (default `None`)
- **project_id** (*int*) – Project id (default `None`)
- **sort_by** (*str*) – Field to sort by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of planning entries

Return type *moco_wrapper.util.response.PagedListResponse*

update(*planning_entry_id: int*, *project_id: int = None*, *starts_on: datetime.date = None*, *ends_on: datetime.date = None*, *hours_per_day: float = None*, *user_id: int = None*, *comment: str = None*, *symbol: moco_wrapper.models.planning_entry.PlanningEntrySymbol = None*)
Updates a planning entry.

Parameters

- **planning_entry_id** (*int*) – Id of the entry to update
- **project_id** (*int*) – Project id (default `None`)
- **starts_on** (*datetime.date, str*) – Start date (default `None`)
- **ends_on** (*datetime.date, str*) – End date (default `None`)
- **hours_per_day** (*float*) – Hours per day the planning entry will consume (default `None`)
- **user_id** (*int*) – User id the planning entry belongs to (default `None`)
- **comment** (*str*) – A comment (default `None`)
- **symbol** (*PlanningEntrySymbol*, int) – Symbol icon to use (default `None`)

Returns The updated planning entry

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.planning_entry.**PlanningEntrySymbol**
Enumeration for allowed values for argument `symbol` for *PlanningEntry.create()*.

```
from moco_wrapper.models.planning_entry import PlanningEntrySymbol
from moco_wrapper import Moco

m = Moco()

new_planning_entry = m.create(
    ..
    symbol = PlanningEntrySymbol.HOME
)
```

BABY_CARRIAGE = 7

BELLS = 6

BUILDING = 2

CAR = 3

COCKTAIL = 5

GRADUATION_CAP = 4

HOME = 1

INFO_CIRCLE = 10

MOON = 9

USERS = 8

5.18 Purchase

class moco_wrapper.models.**Purchase**(*moco*)

create(*purchase_date: datetime.date*, *currency: str*, *payment_method: moco_wrapper.models.purchase.PurchasePaymentMethod*, *items: list*, *due_date: datetime.date = None*, *service_period_from: datetime.date = None*, *service_period_to: datetime.date = None*, *company_id: int = None*, *receipt_identifier: str = None*, *info: str = None*, *iban: str = None*, *reference: str = None*, *custom_properties: dict = None*, *file: moco_wrapper.models.purchase.PurchaseFile = None*, *tags: list = None*)

Create a new purchase.

Parameters

- **purchase_date** (*datetime.date, str*) – Date of the purchase
- **currency** (*str*) – Currency
- **payment_method** (*PurchasePaymentMethod*, str) – Method of payment that was used
- **items** (*List*) – List of items that were purchased (minimum 1 item required)
- **due_date** (*datetime.date, str*) – Date the purchase is due (default None)
- **service_period_from** (*datetime.date, str*) – Service period start date (default None)
- **service_period_to** (*datetime.date, str*) – Service period end date (default None)
- **company_id** (*int*) – Id of the supplier company (default None)
- **receipt_identifier** (*str*) – Receipt string (default None)
- **info** (*str*) – Info text (default None)
- **iban** (*str*) – Iban (default None)
- **reference** (*str*) – Reference text (default None)
- **custom_properties** (*dict*) – Custom Properties (default None)
- **file** (*PurchaseFile*) – File attached to the purchase (default None)

- **tags** (*list*) – List of tags (default None)

Returns The created purchase

Return type *moco_wrapper.util.response.ObjectResponse*

delete (*purchase_id: int*)
Deletes a purchase.

Parameters **purchase_id** (*int*) – Id of the purchase to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

> **Warning:** Deletion of a purchase is only possible if the state of the purchase is *PurchaseStatus.PENDING* and no payments have been registered to the purchase yet

get (*purchase_id: int*)
Retrieve a single purchase.

Parameters **purchase_id** (*int*) – The id of the purchase

Returns A single purchase

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*purchase_id: int = None*, *category_id: int = None*, *term: str = None*, *company_id: int = None*, *status: moco_wrapper.models.purchase.PurchaseStatus = None*, *tags: list = None*, *start_date: datetime.date = None*, *end_date: datetime.date = None*, *unpaid: bool = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of purchases.

Parameters

- **purchase_id** (*int*) – Id of the purchase (default None)
- **category_id** (*int*) – Id of the category the purchase belongs to (default None)
- **term** (*str*) – Full text search (default None)
- **company_id** (*int*) – Company id of the supplier (default None)
- **status** (*PurchaseStatus*, str) – Status of the purchases
- **tags** (*list*) – List of tags (default None)
- **start_date** (*datetime.date, str*) – Start date filter (if start_date is supplied, end_date must also be supplied) (default None)
- **end_date** (*datetime.date, str*) – End date filter (if end_date is supplied, start_date must also be supplied) (default None)
- **unpaid** (*bool*) – Filter only purchases without a payment (default None)
- **sort_by** (*str*) – Field to sort results by (default None)
- **sort_order** (*str*) – asc or desc (default "asc")
- **page** (*int*) – Page number (default 1)

Returns List of purchases

Return type *moco_wrapper.util.response.PagedListResponse*

store_document (*purchase_id*, *file*)
Stores the document for a purchase.

Parameters

- **purchase_id** (*int*) – Id of the purchase
- **file** (*PurchaseFile*) – Purchase file

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

update_status (*purchase_id: int*, *status: moco_wrapper.models.purchase.PurchaseStatus*)
Updates the state of a purchase.

Parameters

- **purchase_id** (*int*) – Id of the purchase to update
- **status** (*PurchaseStatus*, str) – New status

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

class moco_wrapper.models.purchase.**PurchasePaymentMethod**
Enumeration for the allowed values than can be supplied for the `payment_method` argument of *Purchase.create()*.

Example usage:

```
from moco_wrapper.models.purchase import PurchasePaymentMethod
from moco_wrapper import Moco

m = Moco()
new_purchase = m.Purchase.create(
    ..
    payment_method = PurchasePaymentMethod.CASH
)
```

BANK_TRANSFER = 'bank_transfer'

CASH = 'cash'

CREDIT_CARD = 'credit_card'

DIRECT_DEBIT = 'direct_debit'

PAYPAL = 'paypal'

class moco_wrapper.models.purchase.**PurchaseStatus**
Enumeration for the allowed values that can be supplied for the `status` argument of *Purchase.update_status()*.

Example usage:

```
from moco_wrapper.models.purchase import PurchaseStatus
from moco_wrapper import Moco

m = Moco()
m.Purchase.update_status(
    ..
    status = PurchaseStatus.APPROVED
)
```

APPROVED = 'approved'

PENDING = 'pending'

class moco_wrapper.models.purchase.**PurchaseFile**(*file_path*, *file_name=None*)
Helper class for handling files in *Purchase.create()* and *Purchase.store_document()*.

classmethod load(*path*)
Helper method to create a *PurchaseFile* object from a path.

Returns *PurchaseFile* object

Return type *PurchaseFile*

to_base64()
Converts the content of the file to its base64 representation.

Returns File content as base64

Return type str

5.19 PurchaseCategory

class moco_wrapper.models.**PurchaseCategory**(*moco*)
Class for handling Purchase Categories.

get(*category_id: int*)
Retrieve a single category.

Parameters **category_id** (*int*) – Id of the category to retrieve

Returns Single Category object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist()
Retrieve a list of categories.

Returns List of categories

Return type *moco_wrapper.util.response.ListResponse*

5.20 Schedule

class moco_wrapper.models.**Schedule**(*moco*)
Class for handling user absences.

Note: For handling planning, use the *moco_wrapper.models.PlanningEntry*

---

create(*schedule_date: datetime.date*, *absence_code: moco_wrapper.models.schedule.ScheduleAbsenceCode*, *user_id: int = None*, *am: bool = None*, *pm: bool = None*, *comment: str = None*, *symbol: moco_wrapper.models.schedule.ScheduleSymbol = None*, *overwrite: bool = None*)
Create a new schedule entry.

Parameters

- **schedule_date** (*datetime.date, str*) – date of the entry
- **absence_code** (*ScheduleAbsenceCode*, int) – Type of absence

- **user_id** (*int*) – User id (default None)
- **am** (*bool*) – Morning yes/no (default None)
- **pm** (*bool*) – Afternoon yes/no (default None)
- **comment** (*str*) – Comment text (default None)
- **symbol** (*ScheduleSymbol*, int) – Symbol to use for the schedule item (default None)
- **overwrite** (*bool*) – yes/no overwrite existing entry (default None)

Returns The created planning entry

Return type *moco_wrapper.util.response.ObjectResponse*

delete (*schedule_id: int*)
Delete a schedule entry.

Parameters **schedule_id** (*int*) – Id of the entry to delete

Returns The deleted schedule object

Return type *moco_wrapper.util.response.ObjectResponse*

get (*schedule_id: int*)
Retrieve a single schedule object.

Parameters **schedule_id** (*int*) – Id of the entry

Returns Single schedule object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*from_date: datetime.date = None*, *to_date: datetime.date = None*, *user_id: int = None*, *absence_code: moco_wrapper.models.schedule.ScheduleAbsenceCode = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve all planned schedule items.

Parameters

- **from_date** (*datetime.date, str*) – Start date (default None)
- **to_date** (*datetime.date, str*) – End date (default None)
- **user_id** (*int*) – user id the planned entries are belonging to (default None)
- **absence_code** (*ScheduleAbsenceCode*, int) – Type of absence (default None)
- **sort_by** (*str*) – Field to sort the results by (default None)
- **sort_order** (*str*) – asc or desc (default "asc")
- **page** (*int*) – Page number (default 1)

Returns List of schedule objects

Return type *moco_wrapper.util.response.PagedListResponse*

update (*schedule_id: int*, *absence_code: moco_wrapper.models.schedule.ScheduleAbsenceCode = None*, *am: bool = None*, *pm: bool = None*, *comment: str = None*, *symbol: moco_wrapper.models.schedule.ScheduleSymbol = None*, *overwrite: bool = None*)
Update a schedule entry.

Parameters

- **schedule_id** (*int*) – Id of the entry to update
- **absence_code** (*ScheduleAbsenceCode*, int) – Type of absence (default None)

- **am** (*bool*) – Morning yes/no (default None)
- **pm** (*bool*) – Afternoon yes/no (default None)
- **comment** (*str*) – Comment text (default None)
- **symbol** (*ScheduleSymbol*, str) – Symbol to use for the schedule item (default None)
- **overwrite** (*bool*) – yes/no overwrite existing entry (default None)

Returns The updated schedule entry

Return type *moco_wrapper.util.response.ObjectResponse*

class moco_wrapper.models.schedule.**ScheduleAbsenceCode**

Enumeration for allowed values of argument absence_code of *Schedule.getlist()*, *Schedule.create()* and *Schedule.update()*.

```
from moco_wrapper.models.schedule import ScheduleAbsenceCode
from moco_wrapper import Moco

m = Moco()
new_schedule = m.Schedule.create(
    ..
    absence_code = ScheduleAbsenceCode.SICK_DAY
)
```

ABSENCE = 5

HOLIDAY = 4

PUBLIC_HOLIDAY = 2

SICK_DAY = 3

UNPLANNED = 1

class moco_wrapper.models.schedule.**ScheduleSymbol**

Enumeration for allowed values of argument symbol of *Schedule.create()* and *Schedule.update()*.

```
from moco_wrapper.models.schedule import ScheduleSymbol
from moco_wrapper import Moco

m = Moco()
new_schedule = m.Schedule.create(
    ..
    symbol = ScheduleSymbol.HOME
)
```

BABY_CARRIAGE = 7

BELLS = 6

BUILDING = 2

CAR = 3

COCKTAIL = 5

DOT_CIRCLE = 11

EXCLAMATION_MARK = 12

GRADUATION_CAP = 4

HOME = 1

INFO_CIRCLE = 10

MOON = 9

USERS = 8

5.21 Tagging

class moco_wrapper.models.**Tagging**(*moco*)
Class for handling tags/labels.

add(*entity: moco_wrapper.models.tagging.TaggingEntity*, *entity_id: int*, *tags: list*)
Add tags to an entity.

Parameters

- **entity** (*TaggingEntity, str*) – Type of entity to add tags to
- **entity_id** (*int*) – Id of the entity
- **tags** (*list*) – List of tags to add

Returns List of tags assigned to the entity

Return type *moco_wrapper.util.response.ListResponse*

Note: If you supply tags that already exist for the entity they will be ignored.

---

delete(*entity: moco_wrapper.models.tagging.TaggingEntity*, *entity_id: int*, *tags: list*)
Removes supplied tags from an entity

Parameters

- **entity** (*TaggingEntity, str*) – Type of entity to remove tags for
- **entity_id** (*int*) – Id of the entity
- **tags** (*list*) – List of tags to remove

Returns List of tags assigned to the entity

Return type *moco_wrapper.util.response.ListResponse*

get(*entity: moco_wrapper.models.tagging.TaggingEntity*, *entity_id: int*)
Get the tags associated with an entity

Parameters

- **entity** (*TaggingEntity, str*) – Type of entity to add tags to
- **entity_id** (*int*) – Id of the entity

Returns List of tags assigned to the entity

Return type *moco_wrapper.util.response.ListResponse*

replace(*entity: moco_wrapper.models.tagging.TaggingEntity*, *entity_id: int*, *tags: list*)
Replace the tags of an entity.

Parameters

- **entity** (*TaggingEntity, str*) – Type of entity to replace tags for

- **entity_id** (*int*) – Id of the entity
- **tags** (*list*) – New list of tags for the entity

Returns List of tags assigned to the entity

Return type *moco_wrapper.util.response.ListResponse*

Note: You can remove all tags from an entity by supplying an empty list.

class moco_wrapper.models.tagging.**TaggingEntity**

Enumeration for all available entity types that can be supplied for the `entity` argument of *Tagging.add()*, *Tagging.replace()* and *Tagging.delete()*.

Example Usage:

```
from moco_wrapper import Moco
from moco_wrapper.models.tagging import TaggingEntity

m = Moco()

project_id = 22
tags_add = m.Tagging.add(
    entity = TaggingEntity.PROJECT,
    entity_id = project_id,
    tags = [ "these", "are", "the", "tags" ]
)
```

COMPANY = 'Company'

CONTACT = 'Contact'

DEAL = 'Deal'

OFFER = 'Offer'

PROJECT = 'Project'

PURCHASE = 'Invoice'

5.22 Unit

class moco_wrapper.models.**Unit**(*moco*)

Class for handling teams.

When a user is created he always belongs to a team (e.g. development). These can be managed with this model.

get(*unit_id: int*)

Get a single team.

Parameters **unit_id** (*int*) – Id of the team

Returns Single team object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)

Retrieve a list of teams.

Parameters

- **sort_by** (*str*) – Sort by field (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – page number (default `1`)

Returns List of team objects

Return type *moco_wrapper.util.response.PagedListResponse*

5.23 User

class moco_wrapper.models.**User**(*moco*)

Class for handling users.

create(*firstname: str*, *lastname: str*, *email: str*, *password: str*, *unit_id: int*, *active: bool = None*, *external: bool = None*, *language: str = None*, *mobile_phone: str = None*, *work_phone: str = None*, *home_address: str = None*, *birthday: datetime.date = None*, *custom_properties: dict = None*, *info: str = None*)

Creates a new user.

Parameters

- **firstname** (*str*) – First name of the user
- **lastname** (*str*) – Last name of the user
- **email** (*str*) – Email address
- **password** (*str*) – Password to use when creating the user
- **unit_id** (*int*) – Id of the unit/team the user belongs to
- **active** (*bool*) – If the user should be activated or not (default `None`)
- **external** (*bool*) – If the user is an employee or an external employee (his user id will now show up in reports etc.) (default `None`)
- **language** (*str*) – de, de-AT, de-CH, en, it or fr (default `None`)
- **mobile_phone** (*str*) – Mobile phone number (default `None`)
- **work_phone** (*str*) – Work phone number (default `None`)
- **home_address** (*str*) – Physical home address (default `None`)
- **birthday** (*datetime.date, str*) – Birthday date (default `None`)
- **custom_properties** (*dict*) – Custom fields to add to the user (default `None`)
- **info** (*str*) – Additional information about the user (default `None`)

Returns The created user object

Return type *moco_wrapper.util.response.ObjectResponse*

delete(*user_id: int*)

Deletes an existing user.

Parameters **user_id** (*int*) – Id of the user to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get (*user_id: int*)
Get a single user.

Parameters **user_id** (*int*) – Id of the user

Returns Single user object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*include_archived: bool = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Get a list of users.

Parameters

- **include_archived** (*bool*) – Include archived users in the list (default `None`)
- **sort_by** (*str*) – Sort by key (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of users

Return type *moco_wrapper.util.response.PagedListResponse*

update (*user_id*, *firstname: str = None*, *lastname: str = None*, *email: str = None*, *password: str = None*, *unit_id: int = None*, *active: bool = None*, *external: bool = None*, *language: str = None*, *mobile_phone: str = None*, *work_phone: str = None*, *home_address: str = None*, *birthday: datetime.date = None*, *custom_properties: dict = None*, *info: str = None*)
Updates an existing user.

Parameters

- **user_id** (*int*) – the Id of the user
- **firstname** (*str*) – First name of the user (default `None`)
- **lastname** (*str*) – Last name of the user (default `None`)
- **email** (*str*) – Email address (default `None`)
- **password** (*str*) – Password to use when creating the user (default `None`)
- **unit_id** (*int*) – Id of the unit/team the user belongs to (default `None`)
- **active** (*bool*) – If the user should be activated or not (default `None`)
- **external** (*bool*) – If the user is an employee or an external employee (his user id will now show up in reports etc.) (default `None`)
- **language** (*str*) – de, de-AT, de-CH, en, it or fr (default `None`)
- **mobile_phone** (*str*) – Mobile phone number (default `None`)
- **work_phone** (*str*) – Work phone number (default `None`)
- **home_address** (*str*) – Physical home address (default `None`)
- **birthday** (*datetime.date, str*) – Birthday date (default `None`)
- **custom_properties** (*dict*) – Custom fields to add to the user (default `None`)
- **info** (*str*) – Additional information about the user (default `None`)

Returns The updated user object

Return type *moco_wrapper.util.response.ObjectResponse*

5.24 User Employment

class moco_wrapper.models.**UserEmployment**(*moco*)
Class for handling user employment schemes.

Every user has an employment entry, which defines how many hours every day he should be at work.

get(*employment_id: int*)
Retrieve a single employment:

Parameters **employment_id** (*int*) – Id of the employment

Returns Employment object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*from_date: datetime.date = None*, *to_date: datetime.date = None*, *user_id: int = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of employments.

Parameters

- **from_date** (*datetime.date, str*) – Start date (default `None`)
- **to_date** (*datetime.date, str*) – End date (default `None`)
- **user_id** (*int*) – User id (default `None`)
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of employment objects

Return type *moco_wrapper.util.response.PagedListResponse*

5.25 User Holiday

class moco_wrapper.models.**UserHoliday**(*moco*)
Class for handling users holiday/vacation credits.

Every user that can take vacation has a number of vacation credits. These can be manged with this model.

Example usage:

```
from moco_wrapper import Moco

m = Moco()

# this user gets extra vacation in 2020
m.UserHoliday.create(
    year = 2020,
    title = "extra vacation day",
    user_id = 22,
    hours = 8
)
```

Note: Please not that the base unit for holiday credits is hours. So if your typical workday contains 8 hours, 8 holiday credits means one day off.

create (*year: int*, *title: str*, *user_id: int*, *hours: float = None*, *days: float = None*)
Create an users entitlement for holidays/vacation (either by hours or days).

Parameters

- **year** (*int*) – Year the holiday credits are for
- **title** (*str*) – Title
- **user_id** (*int*) – Id of the user these holiday credits belongs to
- **hours** (*float*) – Hours (specify either hours or days) (default `None`)
- **days** (*float*) – Days (specify either hours or days) (default `None`)

Returns The created holiday object

Return type *moco_wrapper.util.response.ObjectResponse*

delete (*holiday_id: int*)
Delete a holiday entry

Parameters **holiday_id** (*int*) – Id of the holiday entry to delete

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get (*holiday_id: int*)
Retrieve single holiday entry

Parameters **holiday_id** (*int*) – Id of the holiday

Returns The holiday object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist (*year: int = None*, *user_id: int = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)
Retrieve a list of holiday entries.

Parameters

- **year** (*int*) – Show only this year (e.g. 2019) (default `None`)
- **user_id** (*int*) – Show only holidays from this user (e.g. 5) (default `None`)
- **sort_by** (*str*) – field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of holiday entries

Return type *moco_wrapper.util.response.PagedListResponse*

update (*holiday_id: int*, *year: int = None*, *title: str = None*, *user_id: int = None*, *hours: float = None*, *days: float = None*)
Update a holiday entry

Parameters

- **holiday_id** (*int*) – Id of the holiday entry

- **year** (*int*) – Year the holiday credits are for (default None)
- **title** (*str*) – Title (default None)
- **user_id** (*int*) – User this holiday entry belongs to (default None)
- **hours** (*float*) – Hours (specify either hours or days) (default None)
- **days** (*float*) – Days (specify either hours or days) (default None)

Returns The updated holiday object

Return type *moco_wrapper.util.response.ObjectResponse*

5.26 User Presence

class moco_wrapper.models.**UserPresence**(*moco*)

Class for handling presences.

With this model you can log the times your user start and finish the workday. For example if the users punch in to indicate they have arrived and punch out to leave, you can log the times with this model.

Also this can be used to log when you users are at work.

create(*pres_date: str*, *from_time: str*, *to_time: str = None*)

Create a presence.

Parameters

- **pres_date** (*datetime.date, str*) – Date of the presence
- **from_time** (*str*) – Starting time of the presence (format HH:MM)
- **to_time** (*str*) – End time of the presence (format HH:MM) (default None)

Returns The created presence

Return type *moco_wrapper.util.response.ObjectResponse*

delete(*pres_id: int*)

Deletes a presence.

Parameters **pres_id** (*int*) – Id of the presence

Returns Empty response on success

Return type *moco_wrapper.util.response.EmptyResponse*

get(*pres_id: int*)

Retrieve a single presence.

Parameters **pres_id** (*int*) – Id of the presence

Returns Single presence object

Return type *moco_wrapper.util.response.ObjectResponse*

getlist(*from_date: datetime.date = None*, *to_date: datetime.date = None*, *user_id: int = None*, *sort_by: str = None*, *sort_order: str = 'asc'*, *page: int = 1*)

Retrieve all user presences.

Parameters

- **from_date** (*datetime.date, str*) – Start date (default None)
- **to_date** (*datetime.date, str*) – End date (default None)

- **user_id** (*int*) – Id of the user (default `None`)
- **sort_by** (*str*) – Field to sort results by (default `None`)
- **sort_order** (*str*) – asc or desc (default `"asc"`)
- **page** (*int*) – Page number (default `1`)

Returns List of presence objects

Return type *moco_wrapper.util.response.PagedListResponse*

Note: `from_date` and `to_date` must be provided together.

---

touch()
Creates a new presence for the user with the corresponding api key starting from the current time. Or it terminates an existing open presence at the current time. Can be used to implement a clock system (RFID).

If a presence is started and stopped within the same minute, it will be discarded.

update(*pres_id: int*, *pres_date: datetime.date = None*, *from_time: str = None*, *to_time: str = None*)
Update a presence.

Parameters

- **pres_id** (*int*) – Id of the presence
- **pres_date** (*datetime.date, str*) – Date of the presence (default `None`)
- **from_time** (*str*) – Starting time of the presence (format HH:MM) (default `None`)
- **to_time** (*str*) – End time of the presence (format HH:MM) (default `None`)

Returns The updated presence

Return type *moco_wrapper.util.response.ObjectResponse*

5.27 Session

class moco_wrapper.models.**Session**(*moco*)
Class for handling authentication against the moco api with a users email address and password.

This model is used internally when the moco instance is created with no api key in the authentication object and will be invoked before the first request is fired.

authenticate(*email: str*, *password: str*)
Authenticates the client with email and password.

Parameters

- **email** (*str*) – Email address
- **password** (*str*) – Password

Returns Authentication information

Return type *moco_wrapper.util.response.ObjectResponse*

CHAPTER 6

Requestor

6.1 Default Requestor

class moco_wrapper.util.requestor.**DefaultRequestor**(*delay_ms: float = 1000.0*)

Default Requestor class that is used by the *moco_wrapper.Moco* instance.

When the default requestor requests a resources and it sees the error code 429 (too many requests), it waits a bit and then tries the request again. If you do not want that behaviour, use *moco_wrapper.util.requestor.NoRetryRequestor*.

See also:

moco_wrapper.util.requestor.NoRetryRequestor

request(*method: str*, *path: str*, *params: dict = None*, *data: dict = None*, *delay_ms: float = 0*, ***kwargs*)

Request the given resource.

Parameters

- **method** (*str*) – HTTP Method (eg. POST, GET, PUT, DELETE)
- **path** (*str*) – Path of the resource (e.g. `/projects`)
- **params** (*dict*) – Url parameters (e.g. `page=1`, query parameters) (default `None`)
- **data** (*dict*) – Dictionary with data (http body) (default `None`)
- **delay_ms** (*float*) – Delay in milliseconds the requestor should wait before sending the request (used for retrying, default `0`)
- **kwargs** – Additional http arguments.

Returns Response object

session

Http Session this requestor uses

6.2 No-Retry Requestor

class moco_wrapper.util.requestor.**NoRetryRequestor**

This requestor works along the same lines as the *moco_wrapper.util.requestor.DefaultRequestor*, but when this requestor comes along the http code 429 for too many requests it just returns an error response.

Use this requestor if you write integration tests or dont have time for retrying the resource.

Example usage:

```
from moco_wrapper.util.requestor import NoRetryRequestor
from moco_wrapper import Moco

no_retry = NoRetryRequestor()
m = Moco(
    requestor = no_retry
)
```

See also:

moco_wrapper.util.requestor.DefaultRequestor

request(*method*, *path*, *params=None*, *data=None*, ***kwargs*)

Request the given resource

Parameters

- **method** (*str*) – HTTP Method (eg. POST, GET, PUT, DELETE)
- **path** (*str*) – Path of the resource (e.g. `/projects`)
- **params** (*dict*) – Url parameters (e.g. `page=1`, query parameters)
- **data** (*dict*) – Dictionary with data (http body)
- **kwargs** – Additional http arguments.

Returns Response object

session

Http Session this requestor uses

6.3 Raw Requestor

class moco_wrapper.util.requestor.**RawRequestor**

The Raw Requestor is a test requestor that saves all arguments into and object and returns it.

> **Warning:** Use this requestor only for testing.

request(*method*, *path*, *params=None*, *data=None*, ***kwargs*) → dict

Request the given resource

Parameters

- **method** – HTTP Method (eg. POST, GET, PUT, DELETE)
- **path** – Path of the resource (e.g. `/projects`)

- **params** – Url parameters (e.g. `page=1`, query parameters)
- **data** – Dictionary with data (http body)
- **kwargs** – Additional http arguments.

Returns Request objects

CHAPTER 7

Objector

7.1 Default Objector

class moco_wrapper.util.objector.**DefaultObjector**

This is the default class for handling the modification ob response objects that the requestor classes generated and were pushed to the objector.

Successful responses will have their data converted into actual python objects, while error responses will be converted into exceptions and raised at a later stage.

Note: If you do not want exceptions to be raised see *moco_wrapper.util.objector.NoErrorObjector*

---

class_map = None

Dictionary used to find the appropriate classes from url-part-path created in *get_class_name_from_request_url()*

For example the path `project=>tasks` means `ProjectTask` is the responsible class. The dictionary contains the following:

```
"projects": {
    "base" => "Project",
    "tasks" => "ProjectTask"
}
```

convert(*requestor_response*)

Converts the data of a response object (for example json) into a python object.

Parameters **requestor_response** – response object (see *Responses*)

Returns modified response object

Note: The data of an error response response (*moco_wrapper.util.response.ErrorResponse*) will be converted into an actual exception that later can be raised

Note: if the method *get_class_name_from_request_url()* that is used to find the right class for conversion, returns `None`, no conversion of objects will take place

error_class_map = None

Dictionary used to convert http status codes into the appropriate exceptions

```
self.error_class_map = {
    404: "NotFoundException",
    ..
}
```

get_class_name_from_request_url (*url*) → str

Finds the class name by analysing the request url.

Parameters **url** (*str*) – url to analyse

This function works as follows:

The url will look something like this `https://test.mocoapp.com/api/v1/projects/1234/tasks?page=1`. We split the url on `/api/v1/`.

```
[https://test.mocoapp.com", "projects/1234/tasks?page=1"]
```

After that we throw away the first part and split the second part on the slash character:

```
["projects", 1234, "tasks?page=1"]
```

Then we remove all query string parameters:

```
["projects", 1234, "tasks"]
```

Then we remove all parts that are ids(digits):

```
["projects", "tasks"]
```

Now that we have our path `projects=>tasks`, we use the *class_map* to find the right classname.

The map is a dictionary that looks something like this:

```
class_map = {
    "activities" => {
        "base" => "Activity"
        "disregard" => None
    },
    "projects": {
        "base" => "Project",
        "tasks" => "ProjectTask"
    },
    "users" => { .. },
    "companies" => { .. }
}
```

We use the path we generated and walk our class_map until we get the entry at the end of the path. In our case that would be `ProjectTask`. As this value is a string that is our final classname.

Note: if the final value is a dictionary, the base case will be returned. For example if path was `projects`, the value at the end of our path is a dictionary. If that is the case the *base* key will be used.

get_error_class_name_from_response_status_code(*status_code*) → str
Get the class name of the exception class based on the given http status code

Parameters **status_code** (*int*) – Http status code of the response

Returns class name of the exception

Warning: The `status_code` parameter must be a key in *error_class_map*

7.2 No-Error Objector

class moco_wrapper.util.objector.**NoErrorObjector**
This class works along the same lines as the DefaultObjector, but it does not convert error responses into actualy exceptions. Instead an ErrorResponse object will be returned.

See also:

moco_wrapper.util.objector.DefaultObjector

Example usage:

```
from moco_wrapper.util.objector import NoErrorObjector
from moco_wrapper import Moco

no_err_objector = NoErrorObjector()
m = Moco(
    objector = no_err_objector
)
```

convert(*requestor_response*)
converts the data of a response object (for example json) into a python object

Parameters **requestor_response** – response object (see *Responses*)

Returns modified response object

Note: only *moco_wrapper.util.response.ObjectResponse* and *moco_wrapper.util.response.PagedListResponse* are object to this conversion. Error responses will not be touched by this objector.

Note: if the method `get_class_name_from_request_url()` that is used to find the right class for conversion, returns `None`, no conversion of objects will take place

7.3 Raw Objector

class moco_wrapper.util.objector.**RawObjector**
Objector class that does no conversion (for testing purposes)

convert(*requestor_response*)
Returns the response object that was given to it.

Parameters **requestor_response** – Response object to convert

Returns requestor_response as it is

CHAPTER 8

Responses

8.1 Object Response

class moco_wrapper.util.response.**ObjectResponse**(*response*)
Class for handling http responses where the body is a single object

data
Returns the json data of the response as a dictionary

```
m = Moco()
project_id = 22

json_response = m.Project.get(project_id).data
print(json_response)
```

response
http response object

8.2 List Response

class moco_wrapper.util.response.**ListResponse**(*response*)
Class for handling responses where the response body is a json list.

The difference to *moco_wrapper.util.response.PagedListResponse* is that ListResponses are not paged.

data
Returns the list of object the response contains

Type list

See also:

items

items
Get the list of objects the response contains

Type list

```
m = Moco()
project_list = m.Project.getlist()

for item in project_list.items:
    print(item)
```

See also:

data

response
http response object

8.3 PagedList Response

class moco_wrapper.util.response.**PagedListResponse**(*response*)
Class for handling http responses where the response body is a json list.

Listings in Moco are usually paginated (if there are more than 100 items that fit your request).

Example usage:

```
from moco_wrapper import Moco

m = Moco()
all_projects = []

project_list = m.Project.getlist()
all_projects.extend(project_list.items)

while not project_list.is_last:
    project_list = m.Project.getlist(page=project_list.next_page)
    all_projects.extend(project_list.items)
```

Or with a for loop:

```
from moco_wrapper import Moco

m = Moco
all_projects = []

project_list = m.Project.getlist()
all_projects.extend(project_list.items)

for i in range(2, project_list.last_page + 1): #first page alread queried
    project_list = m.Project.getlist(page=i)
    all_projects.extend(project_list.items)
```

current_page
Returns the current page number

Type int

data
Returns the list of object the response contains

Type list

See also:

items

is_last
Returns whether the current page is the last page

Type bool

items
Get the list of objects the response contains

Type list

```
m = Moco()
project_list = m.Project.getlist()

for item in project_list.items:
    print(item)
```

See also:

data

last_page
Returns the last page number

Type int

next_page
Returns the next page number

Type int

Note: Do not use this for checking if there is another page, use `is_last()`

page_size
Returns the amount of items that are in the current page (usually 100)

Type int

response
http response object

total
Returns the amount of items that are in the current (paginated) collection

Type int

8.4 File Response

class moco_wrapper.util.response.**FileResponse**(*response*)
Class for handling http responses where the body is just binary content representing a file

data
Returns the binary data of the response

See also:

file

file
Returns the binary data of the response

```
m = Moco()
offer_id = 22
target_path = "./offer.pdf"

file_response = m.Offer.pdf(offer_id)
with open(target_path, "w+b") as f:
    f.write(file_response.file)
```

See also:

data

response
http response object

write_to_file (*file_path: str*)
Writes the binary response content to a file

Parameters **file_path** – path of the target file

```
m = Moco()
offer_id = 22
target_path = "./offer.pdf"

file_response = m.Offer.pdf(offer_id)
file_response.write_to_file(target_path)
```

8.5 Empty Response

class moco_wrapper.util.response.**EmptyResponse**(*response*)
Class for handling responses where the body contains no content but the operation on the api was a success (likely a delete operation)

data
No data in an empty response, returns None

Returns None

response
http response object

8.6 Error Response

class moco_wrapper.util.response.**ErrorResponse**(*response*)
class for handling error messages returned by the api

data
Returns the text of the object (the error message itself)

is_recoverable
Checks the http status code of the response and returns True if the error is not a permanent error, i.e. recovering is possible by simply waiting a bit and sending the request again.

Type bool

Returns `True` if recovery is possible by sending the request again later, otherwise `False`

```
m = Moco()
project_id = 22

project_get = m.Project.get(project_id)

if isinstance(project_get, ErrorResponse) and project_get.is_recoverable:
    time.sleep(5)
    project_get = m.Project.get(project_id)

print(project_get)
```

response
http response object

CHAPTER 9

Generator

9.1 Invoice Payment Generator

class moco_wrapper.util.generator.**InvoicePaymentGenerator**

generate(*payment_date: datetime.date*, *invoice_id: int*, *paid_total: float*, *currency: str*) → dict

Generates an invoice payment item that can be supplied to a bulk created

Parameters

- **payment_date** (*datetime.date, str*) – date of the payment
- **invoice_id** (*int*) – id of the invoice the payment belongs to
- **paid_total** (*float*) – amount that was paid (ex 200)
- **currency** (*str*) – currency of the amout that was paid (ex. EUR)

Returns an invoice payment item

Example usage:

```
from moco_wrapper.util.generator import InvoicePaymentGenerator
from moco_wrapper import Moco
from datetime import date

m = Moco()
gen = InvoicePaymentGenerator()

items = [
    gen.generate(
        "2019-10-10",
        1,
        200,
        "EUR"
    ),
```

(continues on next page)

(continued from previous page)

```
    gen.generate(
        "2020-04-04",
        2,
        540,
        "CHF"
    ),
    gen.generate(
        date(2020, 1, 1)
        1,
        300,
        "EUR"
    )
]

created_invoice_payment = m.InvoicePayment.create_bulk(items)
```

See also:

moco_wrapper.models.InvoicePayment.create_bulk()

9.2 Invoice Item Generator

class moco_wrapper.util.generator.**InvoiceItemGenerator**

generate_description(*description: str*) → dict
Generate an item of type `description`

Parameters **description** (*str*) – Description the item should have

Returns The generated item

generate_detail_position(*title: str*, *quantity: float*, *unit: str*, *unit_price: float*, *activity_ids: list = None*, *expense_ids: list = None*) → dict
Generates a detailed position item to be used in an offer items list (for example hours are a perfect example that can be split into units (a single hours set the unit, unit_price, and quantity)).

Parameters

- **title** (*str*) – Title of the position item
- **quantity** (*float*) – How many of the things (i.e. how many hours)
- **unit** (*str*) – What is the thing (i.e. hours)
- **unit_price** (*float*) – Price of a single thing (i.e. price of a single hour)
- **activity_ids** (*list*) – Ids of the activities billed by this item
- **expense_ids** (*list*) – Ids of the expenses billed by this item

Returns The generated item

See also:

generate_item()

generate_item(*title: str*, *quantity: float = None*, *unit: str = None*, *unit_price: float = None*, *net_total: float = None*, *activity_ids: list = None*, *expense_ids: list = None*) → dict
Generate an invoice item.

Parameters

- **title** (*str*) – Title of the item
- **quantity** (*float*) – Quantity of the supplied item
- **unit** (*str*) – Unit name of the supplied item
- **unit_price** (*float*) – Unit price of the supplied item
- **net_total** (*float*) – Net total sum (either this is supplied or unit, unit_price, and quantity)
- **activity_ids** (*list*) – Ids of the activities billed by this item
- **expense_ids** (*list*) – Ids of the expenses billed by this item

Returns The generated item

This is the base function for generating positions in an invoice. There are two types of positions. Postions that can be itemized (see `generate_detail_postion()`) and positions that do not have to be itemized (*generate_lump_position()*).

See also:

`generate_detail_postion()` and *generate_lump_position()*

generate_lump_position(*title: str*, *net_total: float*) → dict

Generates a general position item to be used in a offer list (use this if the postion cannot (or do not want) to split the position into units).

Parameters

- **title** (*str*) – Title of the position
- **net_total** (*float*) – Total price of the postion

Returns The generated item

See also:

generate_item()

generate_pagebreak() → dict

Generate an item of type `page-break`

Returns The generated item

generate_separator() → dict

Generate an item of type `separator`

Returns The generated item

generate_subtotal(*title: str*) → dict

Generate an item of type `subtotal`

Parameters **title** (*str*) – The title of the subtotal

Returns The generated item

generate_title(*title: str*) → dict

Generate an item of type `title`

Parameters **title** (*str*) – Title the item should have

Returns The generated item

9.3 Offer Item Generator

class moco_wrapper.util.generator.**OfferItemGenerator**

generate_description(*description: str*) → dict
Generate an item of type `description`

Parameters **description** (*str*) – Description the item should have

Returns The generated item

generate_detail_position(*title: str*, *quantity: float*, *unit: str*, *unit_price: float*, *optional: bool = False*) → dict
Generates a detailed position item to be used in an offer items list (for example work hours are a perfect example that can be split into units (a single hours set the unit, unit_price, and quantity))

Parameters

- **title** (*str*) – Title of the position item
- **quantity** (*float*) – How many of the things (i.e. how many hours)
- **unit** (*str*) – What is the thing (i.e. hours)
- **unit_price** (*float*) – Price of a single thing (i.e. price of a single hour)
- **optional** (*bool*) – If the position is optional or not (default False)

Returns The generated item

See also:

generate_item()

generate_item(*title: str*, *quantity: float = None*, *unit: str = None*, *unit_price: float = None*, *net_total: float = None*, *optional=False*) → dict
Generate an offer item

Parameters

- **title** (*str*) – Title of the item
- **quantity** (*float*) – Quantity of the supplied item
- **unit** (*str*) – Unit name of the supplied item
- **unit_price** (*float*) – Unit price of the supplied item
- **net_total** (*float*) – Net total sum (either this is supplied or unit, unit_price, and quantity)
- **optional** (*bool*) – Whether the item is an optional item or not (default False)

Returns The generated item

This is the base function for generating positions in an offer. There are two types of positions. Postions that can be itemized (see `generate_detail_postion()`) and positions that do not have to be itemized (*generate_lump_position()*).

See also:

`generate_detail_postion()` and *generate_lump_position()*

generate_lump_position(*title: str*, *net_total: float*, *optional: bool = False*) → dict
Generates a general position item to be used in a offer list (use this if the postion cannot (or do not want) to split the position into units)

Parameters

- **title** (*str*) – Title of the position
- **net_total** (*float*) – Total price of the postion
- **optional** (*bool*) – If the position is optional or not (default False)

Returns The generated item

See also:

generate_item()

generate_pagebreak() → dict
Generate an item of type `page-break`

Returns The generated item

generate_separator() → dict
Generate an item of type `separator`

Returns The generated item

generate_subtotal(*title: str*) → dict
Generate an item of type `subtotal`

Parameters **title** (*str*) – The title of the subtotal

Returns The generated item

generate_title(*title: str*) → dict
Generate an item of type `title`

Parameters **title** (*str*) – Title the item should have

Returns The generated item

9.4 Project Expense Generator

class moco_wrapper.util.generator.**ProjectExpenseGenerator**

generate(*expense_date: datetime.date*, *title: str*, *quantity: float*, *unit: str*, *unit_price: float*, *unit_cost: float*, *description: str = None*, *billable: bool = True*, *budget_relevant: bool = False*) → dict
Create an item that can be used for creating bulk project expenses.

Parameters

- **project_id** (*int*) – Id of the project to create the expense for
- **expense_date** (*datetime.date, str*) – Date of the expense
- **title** (*str*) – Exense title
- **quantity** (*float*) – Quantity (how much of `unit` was bought?)
- **unit** (*str*) – Name of the unit (What was bought for the customer/project?)
- **unit_price** (*float*) – Price of the unit that will be billed to the customer/project
- **unit_cost** (*float*) – Cost that we had to pay
- **description** (*str*) – Descripion of the expens
- **billable** (*bool*) – If this expense billable (default True)

- **budget_relevant** (*bool*) – If this expense is budget relevant (default False)

Returns The created expense object

Example usage:

```
from moco_wrapper.util.generator import ProjectExpenseGenerator
from moco_wrapper import Moco
from datetime import date

m = Moco()
gen = ProjectExpenseGenerator()

items = [
    gen.generate(
        '2019-10-10',
        "the title",
        5,
        "the unit",
        20,
        10
    ),
    gen.generate(
        '2019-10-10',
        "different title",
        5,
        "another unit",
        20,
        10,
        billable=False,
        description="the desc",
        budget_relevant=True
    ),
    gen.generate(
        date(2019, 10, 10),
        "another title",
        2,
        "the unit",
        20,
        10
    ),
]
project_id = 2

created_expenses = m.ProjectExpense.create_bulk(project_id,items)
```

See also:

moco_wrapper.models.ProjectExpense.create_bulk()

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

Y