

---

# **moco-wrapper Documentation**

***Release 0.6.0***

**sommalia**

**Mar 18, 2020**



---

## Getting Started

---

<b>1</b>	<b>Quick Start</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Authentication</b>	<b>7</b>
<b>4</b>	<b>Impersonation</b>	<b>9</b>
<b>5</b>	<b>The Moco Instance</b>	<b>11</b>
<b>6</b>	<b>Requestor</b>	<b>53</b>
<b>7</b>	<b>Objector</b>	<b>57</b>
<b>8</b>	<b>Responses</b>	<b>61</b>
<b>9</b>	<b>Generator</b>	<b>65</b>
	<b>Index</b>	<b>71</b>



This documentation is split into the following parts:

- *Getting Started*
- *Code Overview*



# 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 application 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)
<ListingResponse, 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 impersonate 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 `impersonate` 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().*

---

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 resources 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()*

#### **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
```

#### **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*

#### **request(method, path, params=None, data=None, bypass\_auth=False)**

request 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:

```
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 (see project tasks)
- **hours** (float) – hours to log to the activity (passing a 0 will start a timer if the date is today)
- **description** (str) – activity description text
- **billable** (bool) – true/false (if this activity is billable) (select none if billing is dependent on project configuration)
- **tag** (str) – a tag string
- **remote\_service** (ActivityRemoteService, str) – if this task was created by a remote service, its name will be here.

- **remote\_id** (*str*) – id of the activity in the remote\_service
- **remote\_url** (*str*) – address of the remote service

**Returns** the created activity

**delete** (*activity\_id: int*)

Delete an activity.

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

**Returns** empty response on success

**disregard** (*reason, activity\_ids, company\_id, project\_id=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

**Returns** List with the activity ids that were disregarded

**get** (*activity\_id: int*)

Get a single activity.

**Parameters** **activity\_id** (*int*) – Id of the activity:

**Returns** The activity object

**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 activity objects.

**Parameters**

- **from\_date** (*datetime.date, str*) – Start date
- **to\_date** (*datetime.date, str*) – End date
- **user\_id** (*int*) – User id of the creator
- **project\_id** (*int*) – Id of the project the activity belongs to
- **task\_id** (*int*) – Id of the task the activity belongs to
- **sort\_by** (*str*) – Field to sort results by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – Page number (default 1)

**Returns** List of activities

**start\_timer** (*activity\_id: int*)

Start a timer on the specified activity.

The timer can only be started for activities on the current day.

**Parameters** **activity\_id** (*int*) – id of the activity

**Returns** the activity the timer was started for

**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

**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`)  
Create 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 (see project tasks)
- **hours(float)** – hours to log to the activity (passing a 0 will start a timer if the date is today)
- **description(str)** – activity description text
- **billable(bool)** – true/false (if this activity is billable) (select none if billing is dependent on project configuration)
- **tag(str)** – a tag string
- **remote\_service(ActivityRemoteService, str)** – if this task was created by a remote service, its name will be here
- **remote\_id(str)** – id of the activity in the remote\_service
- **remote\_url(str)** – address of the remote service

**Returns** the updated activity

**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'

**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 mean (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

**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. For allowed values see [CommentTargetType](#).
- **text** (*str*) – comment text

**Returns** List of created comments.

**delete** (*comment\_id: int*)  
Delete a comment.

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

**Returns** Empty response on success

**get** (*comment\_id: int*)  
Retrieve a single comment.

**Parameters** **comment\_id** (*int*) – Id of the comment

**Returns** Single comment

**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
- **commentable\_id** (*int*) – Id of the object the comment belongs to
- **user\_id** (*int*) – User id of the creator
- **manual** (*bool*) – If the comment was user-created or generated
- **sort\_by** (*str*) – Field to sort the results by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – Page number (default 1)

**Returns** list of comments

**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

**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
)
```

**CONTACT** = 'Contact'

**CUSTOMER** = 'Customer'

**DEAL** = 'Deal'

**INVOICE** = 'Invoice'

**OFFER** = 'Offer'

**OFFERCONFIRMATION** = 'OfferConfirmation'

```
PROJECT = 'Project'
```

```
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, 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)

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
- **fax** (str) – Fax number of the company
- **phone** (str) – Phone number of the company
- **email** (str) – Email address of the company
- **info** (str) – Additional information about the company
- **custom\_properties** (dict) – Custom properties dictionary
- **labels** (list) – Array of labels
- **user\_id** (int) – User Id of the responsible person
- **currency** (str) – Currency the company uses (mandatory when company\_type == customer)
- **identifier** (str) – Identifier of the company (only mandatory when not automatically assigned)
- **billing\_tax** (float) – Billing tax value (from 0 to 100)

- **default\_invoice\_due\_days** (*int*) – Payment target days for the company when creating invoices
- **country\_code** (*str*) – ISO Alpha-2 Country Code like “DE” / “CH” / “AT” in upper case - default is account country
- **vat\_identifier** (*str*) – Vat identifier for eu companies (only supplier and customer)
- **iban** (*str*) – Iban number (only supplier)

**Returns** The created company

**get** (*comany\_id: int*)

Get a single company.

**Parameters** **comany\_id** (*int*) – Id of the company

**Returns** Single company object

**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
- **tags** (*list*) – List of tags
- **identifier** (*str*) – Company identifier
- **sort\_by** (*str*) – Field to sort by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – page number (default 1)

**Returns** list of companies

**update** (*comany\_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, 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*)

Update a company.

**Parameters**

- **comany\_id** (*int*) – Id of the company
- **company\_type** (*CompanyType*, *str*) – Type of the company to modify
- **name** (*str*) – Name of the company
- **website** (*str*) – Url of the companies website
- **fax** (*str*) – Fax number of the company
- **phone** (*str*) – Phone number of the company
- **email** (*str*) – Email address of the company
- **info** (*str*) – Additional information about the company
- **custom\_properties** (*dict*) – Custom properties dictionary
- **labels** (*list*) – Array of labels

- **user\_id** (*int*) – Id of the responsible person
- **currency** (*str*) – Currency the company uses (only mandatory when company\_type == customer)
- **identifier** (*str*) – Identifier of the company (only mandatory when not automatically assigned)
- **billing\_tax** (*float*) – Billing tax value
- **default\_invoice\_due\_days** (*int*) – payment target days for the company when creating invoices
- **country\_code** (*str*) – ISO Alpha-2 Country Code like “DE” / “CH” / “AT” in upper case - default is account country
- **vat\_identifier** (*str*) – vat identifier for eu companies (only supplier and customer)
- **iban** (*str*) – iban number (only supplier)

**Returns** The updated company

**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
- **title** (*str*) – Job title the contact has
- **job\_position** (*str*) – Name of the job position this contact has
- **mobile\_phone** (*str*) – Mobile phone number the contact has
- **work\_fax** (*str*) – Work fax number
- **work\_phone** (*str*) – Work phone number
- **work\_email** (*str*) – Work email address
- **work\_address** (*str*) – Physical work address
- **home\_address** (*str*) – Physical home address
- **home\_email** (*str*) – Home email address
- **birthday** (*datetime.date*, *str*) – Birthday date
- **info** (*str*) – More information about the contact
- **tags** (*list*) – Array of additional tags

**Returns** The created contact object

**get** (*contact\_id: int*)

Retrieve a single contact object

**Parameters** **contact\_id** (*int*) – Id of the contact

**Returns** The contact object

**getlist** (*tags: list = None, sort\_by: str = None, sort\_order: str = 'asc', page: int = 1*)

Retrieve a list of contact objects

**Parameters**

- **tags** (*list*) – Array of tags
- **sort\_by** (*str*) – Field to the results by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – Page number (default 1)

**Returns** List of contact objects

**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
- **lastname** (*str*) – The last name of the contact
- **gender** (*ContactGender*, *str*) – Gender of the contact. For allowed values see [ContactGender](#).

- **company\_id** (*int*) – Id of the company the contact belongs to
- **title** (*str*) – Job title the contact has
- **job\_position** (*str*) – name of the job position this contact has
- **mobile\_phone** (*str*) – Mobile phone number the contact has
- **work\_fax** (*str*) – Work fax number
- **work\_phone** (*str*) – Work phone number
- **work\_email** (*str*) – Work email address
- **work\_address** (*str*) – Physical work address
- **home\_address** (*str*) – Physical home address
- **home\_email** (*str*) – Home email address
- **birthday** (*datetime.date, str*) – Birthday date
- **info** (*str*) – More information about the contact
- **tags** (*list*) – Array of additional tags

**Returns** The updated contact object

**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: str = '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
- **info** (*str*) – Additional information
- **status** (*DealStatus*, *str*) – Current state of the deal

**Returns** The created deal object

**get** (*deal\_id: int*)

Retrieve a single deal.

**Parameters** **deal\_id** (*int*) – Id of the deal

**Returns** Single deal object

**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
- **tags** (*list*) – Array of tags
- **sort\_by** (*str*) – Field to order results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of deal objects

**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
- **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 that is responsible for this deal
- **deal\_category\_id** (*int*) – Deal category id
- **company\_id** (*int*) – Company id
- **info** (*str*) – Additional information
- **status** (*DealStatus*, *str*) – Current state of the deal

**Returns** The updated deal object

**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).

Typicly 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

**delete** (*category\_id: int*)

Delete a deal category.

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

**Returns** Empty response on success

**get** (*category\_id: int*)

Retrieves a single deal category.

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

**Returns** Single deal category

**getlist** (*sort\_by: str = None, sort\_order: str = 'asc', page: int = 1*)

Retrieves a list of a deal categories.

**Parameters**

- **sort\_by** (*str*) – Field to sort by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of deal categories

**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)
- **probability** (*int*) – Deal category success probability (between 1 and 100)

**Returns** The updated deal category

## 5.7 Invoice

**class** `moco_wrapper.models.Invoice` (*moco*)

Models 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*)

Creates a new invoice.

**Parameters**

- **customer\_id** (*int*) – Id of the customer/company
- **recipient\_address** (*str*) – Entry text for the customer (e.g. “My Customer\nMainStreet 5\nExample Town”)
- **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: “created”)
- **change\_address** (*InvoiceChangeAddress*, *str*) – Address propagation (default: “invoice”)
- **salutation** (*str*) – Salutation text
- **footer** (*str*) – Footer text
- **discount** (*float*) – Discount in percent (between 0.0 and 100.0)
- **cash\_discount** (*float*) – Cash discount in percent (between 0.0 and 100.0)
- **cash\_discount\_days** (*float*) – How many days is the cash discount valid (ex. 4)
- **project\_id** (*int*) – Id of the project the invoice belongs to

**Returns** The created invoice

---

**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

**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
- **date\_from** (*datetime.date*, *str*) – Starting date
- **date\_to** (*datetime.date*, *str*) – End date
- **tags** (*list*) – List of tags
- **identifier** (*str*) – Identifier string (e.g. R1903-003)
- **term** (*str*) – Wildcard search term
- **sort\_by** (*str*) – Field to sort results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of invoice objects

**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
- **date\_from** (*datetime.date*, str) – Start date
- **date\_to** (*datetime.date*, str) – End date
- **identifier** (str) – Identifier string (ex. R1903-003)
- **sort\_by** (str) – Field to sort results by
- **sort\_order** (str) – asc or desc (default asc)
- **page** (int) – Page number (default 1)

**Returns** List of invoice objects

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

**Parameters** **invoice\_id** (int) – Invoice id

**Returns** Invoice pdf

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

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

**Parameters** **invoice\_id** (int) – Invoice id

**Returns** Invoice timesheet as pdf

**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

**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 this status 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'
```

## 5.8 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 (ex. 193.50)
- **currency** (*str*) – Currency used (e.g. EUR)

**Returns** The created invoice payment object

**create\_bulk** (*items: list = []*)

Create multiple new invoice payments.

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



**Returns** List of created invoice payments

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

**get** (*payment\_id: int*)

Retrieve a single invoice payment.

**Parameters** **payment\_id** (*int*) – Invoice payment id

**Returns** Single invoice payment object

**getlist** (*invoice\_id: int = None, date\_from: datetime.date = None, date\_to: datetime.date = None, sort\_by: str = None, sort\_order='asc', page=1*)

Retrieve a list of invoice payments.

**Parameters**

- **invoice\_id** (*int*) – Id of a corresponding invoice
- **date\_from** (*datetime.date, str*) – Start date
- **date\_to** (*datetime.date, str*) – End date
- **sort\_by** (*str*) – Field to sort results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of invoice payments

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

**Returns** The updated invoice payment object

## 5.9 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 (e.g. Test Custmer\nMain Street 5\nExample Town)
- **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
- **salutation** (*str*) – Salutation text
- **footer** (*str*) – Footer text
- **discount** (*float*) – Discount in percent
- **contact\_id** (*int*) – Id of the contact for the offer

**Returns** The created offer

---

**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

**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
- **from\_date** (*datetime.date*, str) – Start date
- **to\_date** (*datetime.date*, str) – End date
- **identifier** (str) – Identifier string (e.g.: “A1903-003”)
- **sort\_by** (str) – Field to sort the results by
- **sort\_order** (str) – asc or desc (default asc)
- **page** (int) – Page number (default 1)

**Returns** List of offer objects

---

**Note:** Offers can be sorted by date, created\_at and title.

---

**pdf** (*offer\_id: int*, *letter\_paper\_id: int = None*)  
 Retrieve the offer document for a single offer.

**Parameters**

- **offer\_id** (int) – Id of the offer
- **letter\_paper\_id** (int) – Id of the letter paper (default white)

**Returns** The offers pdf document

**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'

## 5.10 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

**assigned** (*active: bool = None, sort\_by: str = None, sort\_order: str = 'asc', page: int = 1*)

Get list of all project currently assigned to the user.

**Parameters**

- **active** (*bool*) – Show only active or inactive projects
- **sort\_by** (*str*) – Sort by field
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of project objects

**create** (*name: str, currency: str, leader\_id: int, customer\_id: int, finish\_date: datetime.date = None, identifier: str = None, billing\_address: str = 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
- **finish\_date** (*datetime.date, str*) – Finish date
- **identifier** (*str*) – Project Identifier
- **billing\_address** (*str*) – Billing address the invoices go to

- **billing\_variant** (*ProjectBillingVariant*, str) – Billing variant used
- **hourly\_rate** (*float*) – Hourly rate that will get billed
- **budget** (*float*) – Budget for the project
- **labels** (*list*) – Array of additional labels
- **custom\_properties** (*dict*) – Custom values used by the project
- **info** (*str*) – Additional information
- **fixed\_price** (*bool*) – If the project is a fixed price projects (default False)

**Returns** The created project object

---

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

---



---

**Note:** If you create a `fixed_price` project budget must also be set.

---

**get** (*project\_id: int*)  
Get a single project.

**Parameters** **project\_id** (*int*) – Id of the project

**Returns** Project object

**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
- **include\_company** (*bool*) – Include the complete company object or just the company id
- **leader\_id** (*int*) – User id of the project leader
- **company\_id** (*int*) – Company id the projects are assigned to
- **created\_from** (*datetime.date, str*) – Created start date
- **created\_to** (*datetime.date, str*) – Created end date
- **updated\_from** (*datetime.date, str*) – Updated start date
- **updated\_to** (*datetime.date, str*) – Updated end date
- **tags** (*list*) – Array of tags
- **identifier** – Project identifier
- **sort\_by** (*str*) – Field to sort the results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of project objects

**report** (*project\_id: int*)

Retrieve a project report.

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

**Parameters** **project\_id** (*int*) – Id of the project

**Returns** Report with the most important project business indicators

**unarchive** (*project\_id: int*)

Unarchive a project.

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

**Returns** The unarchived project

**update** (*project\_id: int, name: str = None, leader\_id: int = None, customer\_id: int = None, finish\_date: datetime.date = None, identifier: str = None, billing\_address: str = 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
- **leader\_id** (*int*) – User id of the project leader
- **customer\_id** (*int*) – Company id of the customer
- **finish\_date** (*datetime.date, str*) – Finish date
- **identifier** (*str*) – Project Identifier
- **billing\_address** (*str*) – Address the invoices go to
- **billing\_variant** (*ProjectBillingVariant, str*) – Billing variant used
- **hourly\_rate** (*float*) – Hourly rate that will get billed
- **budget** (*float*) – Budget for the project
- **labels** (*list*) – Array of additional labels
- **custom\_properties** (*dict*) – Custom values used by the project
- **info** (*str*) – Additional information

**Returns** The updated project object

**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'
```

## 5.11 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
- **active** (*bool*) – If the contract is active
- **budget** (*float*) – Contract budget
- **hourly\_rate** (*float*) – Contract hourly rate

**Returns** Created contract object

**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

**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

**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

- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of contract objects

**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** – Id of the contract to update
- **billable** (*bool*) – If the contract is billable
- **active** (*bool*) – If the contract is active
- **budget** (*float*) – Contract budget
- **hourly\_rate** (*float*) – Contract hourly rate

**Returns** The updated project contract

## 5.12 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
- **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



**Returns** The created expense object

**create\_bulk** (*project\_id: int, items: list*)  
Create an 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

**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

**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** Empty response on success

Example usage:

```
from moco_wrapper import Moco

m = Moco()
project_id = 22
expense_ids_to_disregard = [444, 522, 893]

m.ProjectExpense.disregard(
    project_id,
    expense_ids_to_disregard,
    "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

**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
- **to\_date** (*datetime.date, str*) – End date
- **sort\_by** (*str*) – Sort results by field
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of expense objects

**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
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of expense objects

**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
- **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
- **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

**Returns** The updated expense object

## 5.13 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.ProjectRecurringExpensePeriod, 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 empty: unlimited)
- **description** (*str*) – Description of the expense
- **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

**Returns** The created recurring expense object

**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

**get** (*project\_id: int, recurring\_expense\_id: int*)

Retrieve a single recurring expense

### Parameters

- **project\_id** (*int*) – Id of the project the expesen belongs to
- **recurring\_expense\_id** (*int*) – id of the recurring expense

**Returns** Single recurring expense object

**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 expesen belongs to
- **sort\_by** (*str*) – Field to sort results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of recurring expenses

**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
- **quantity** (*int*) – 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 empty: unlimited)
- **description** (*str*) – Description of the expense
- **billable** (*bool*) – If this expense billable
- **custom\_properties** (*dict*) – Additional fields as dictionary

**Returns** The updated recurring expense object

**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'

```

## 5.14 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()
leader_id = 1
customer_id = 2

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

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

- **checked** – Mark entry as checked (the entry will be crossed out in the UI)

**Returns** The created schedule item

**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

**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

**getlist** (*project\_id: int, sort\_by: str = None, sort\_order: str = 'asc', page: int = 1*)

Retrieve a list of project payment schedules

**Parameters**

- **project\_id** (*int*) – Project id of the schedule items
- **sort\_by** (*str*) – Field to sort the results by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – Page number (default 1)

**Returns** List of schedules payments

**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
- **schedule\_date** (*datetime.date, str*) – Date the billing will take place
- **title** (*str*) – Title of the item
- **checked** (*bool*) – Mark entry as checked (the entry will be crossed out in the UI)

**Returns** The updated schedule item

## 5.15 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)
- **hourly\_rate** (*float*) – How much is the hourly rate for the task (e.g.: 120.5)

**Returns** The created project task

**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

---

**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

**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
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of project tasks

**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
- **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)
- **hourly\_rate** (*float*) – How much is the hourly rate for the task (e.g.: 120.5)

**Returns** The updated project task

## 5.16 Schedule

**class** `moco_wrapper.models.Schedule` (*moco*)

Class for handling user schedules.

**create** (*schedule\_date: datetime.date, project\_id: int = None, absence\_code: moco\_wrapper.models.schedule.ScheduleAbsenceCode = None, 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
- **project\_id** (*int*) – Project id
- **absence\_code** (*ScheduleAbsenceCode, int*) – Type of absence
- **user\_id** (*int*) – User id
- **am** (*bool*) – Morning yes/no
- **pm** (*bool*) – Afternoon yes/no
- **comment** (*str*) – Comment text
- **symbol** (*ScheduleSymbol, int*) – Symbol to use for the schedule item
- **overwrite** (*bool*) – yes/no overwrite existing entry

**Returns** The created planning entry

---

**Note:** Define either `project_id` OR `absence_code`, specify one, not both.

---

**delete** (*schedule\_id: int*)

Delete a schedule entry.

**Parameters** **schedule\_id** (*int*) – Id of the entry to delete

**Returns** Empty response on success

**get** (*schedule\_id: int*)

Retrieve a single schedule object.

**Parameters** **schedule\_id** (*int*) – Id of the entry

**Returns** Single schedule object



```

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

```

Retrieve all planned schedule items.

#### Parameters

- **from\_date** (*datetime.date*, *str*) – Start date
- **to\_date** (*datetime.date*, *str*) – End date
- **user\_id** (*int*) – user id the planned entries are belonging to
- **project\_id** (*int*) – project id
- **absence\_code** (*ScheduleAbsenceCode*, *int*) – Type of absence
- **sort\_by** (*str*) – Field to sort the results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of schedule objects

```

update (schedule_id: int, project_id: int = None, 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
- **project\_id** (*int*) – Project id
- **absence\_code** (*ScheduleAbsenceCode*, *int*) – Type of absence
- **am** (*bool*) – Morning yes/no
- **pm** (*bool*) – Afternoon yes/no
- **comment** (*str*) – Comment text
- **symbol** (*ScheduleSymbol*, *str*) – Symbol to use for the schedule item
- **overwrite** (*bool*) – yes/no overwrite existing entry

**Returns** The updated schedule entry

---

**Note:** Define either `project_id` OR `absence_code`, specify one, not both.

---

**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(
    ..

```

(continues on next page)

(continued from previous page)

```
        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
```

```
GRADUATION_CAP = 4
```

```
HOME = 1
```

```
INFO_CIRCLE = 10
```

```
MOON = 9
```

```
USERS = 8
```

## 5.17 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

**getlist** (*sort\_by: str = None, sort\_order: str = 'asc', page: int = 1*)

Retrieve a list of teams.

#### Parameters

- **sort\_by** (*str*) – Sort by field
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – page number (default 1)

**Returns** List of team objects

## 5.18 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
- **external** (*bool*) – If the user is an employee or an external employee (his user id will now show up in reports etc.)
- **language** (*str*) – de, de-AT, de-CH, en, it or fr
- **mobile\_phone** (*str*) – Mobile phone number
- **work\_phone** (*str*) – Work phone number
- **home\_address** (*str*) – Physical home address
- **birthday** (*datetime.date, str*) – Birthday date
- **custom\_properties** (*dict*) – Custom fields to add to the user
- **info** (*str*) – Additional information about the user

**Returns** The created user object

**delete** (*user\_id: int*)

Deletes an existing user.

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

**Returns** Empty response on success

**get** (*user\_id: int*)

Get a single user.

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

**Returns** Single user object

**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
- **sort\_by** (*str*) – Sort by key
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of users

**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
- **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
- **external** (*bool*) – If the user is an employee or an external employee (his user id will now show up in reports etc.)
- **language** (*str*) – de, de-AT, de-CH, en, it or fr
- **mobile\_phone** (*str*) – Mobile phone number
- **work\_phone** (*str*) – Work phone number
- **home\_address** (*str*) – Physical home address
- **birthday** (*datetime.date, str*) – Birthday date
- **custom\_properties** (*dict*) – Custom fields to add to the user
- **info** (*str*) – Additional information about the user

**Returns** The updated user object

## 5.19 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

**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
- **to\_date** (*datetime.date, str*) – End date
- **user\_id** (*int*) – User id
- **sort\_by** (*str*) – Field to sort results by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – Page number (default 1)

**Returns** List of employment objects

## 5.20 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 managed with this model.

Example usage:

```
from moco_wrapper import Moco

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

**Note:** Please note 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)

- **days** (*float*) – Days (specify either hours or days)

**Returns** The created holiday object

**delete** (*holiday\_id: int*)

Delete a holiday entry

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

**Returns** Empty response on success

**get** (*holiday\_id: int*)

Retrieve single holiday entry

**Parameters** **holiday\_id** (*int*) – Id of the holiday

**Returns** The holiday object

**getlist** (*year: int = None, user\_id: int = None, sort\_by: str = None, sort\_order: str = 'asc', page: int = 1*)

Retrieve a list of holidays entries

**Parameters**

- **year** (*int*) – Show only this year (e.g. 2019)
- **user\_id** (*int*) – Show only holidays from this user (e.g. 5)
- **sort\_by** (*str*) – field to sort results by
- **sort\_order** (*str*) – asc or desc
- **page** (*int*) – Page number (default 1)

**Returns** List of holiday entries

**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
- **title** (*str*) – Title
- **user\_id** (*int*) – User this holiday entry belongs to
- **hours** (*float*) – Hours (specify either hours or days)
- **days** (*float*) – Days (specify either hours or days)

**Returns** The updated holiday object

## 5.21 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)

**Returns** The created presence

**delete** (*pres\_id: int*)

Deletes a presence.

**Parameters** **pres\_id** (*int*) – Id of the presence

**Returns** Empty response on success

**get** (*pres\_id: int*)

Retrieve a single presence.

**Parameters** **pres\_id** (*int*) – Id of the presence

**Returns** Single presence object

**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
- **to\_date** (*datetime.date, str*) – End date
- **user\_id** (*int*) – Id of the user
- **sort\_by** (*str*) – Field to sort results by
- **sort\_order** (*str*) – asc or desc (default asc)
- **page** (*int*) – Page number (default 1)

**Returns** List of presence objects

---

**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
- **from\_time** (*str*) – Starting time of the presence (format HH:MM)
- **to\_time** (*str*) – End time of the presence (format HH:MM)

**Returns** The created presence

## 5.22 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 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



## 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 requests requests a ressource 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, path, params=None, data=None, delay\_ms=0, \*\*kwargs*)

Request the given ressource

### Parameters

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

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 ressource

**Parameters**

- **method** (*str*) – HTTP Method (eg. POST, GET, PUT, DELETE)
- **path** (*str*) – Path of the ressource (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 an object and returns it.

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

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

Request the given ressource

**Parameters**

- **method** – HTTP Method (eg. POST, GET, PUT, DELETE)
- **path** – Path of the ressource (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



## 7.1 Default Objector

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

This is the default class for handling the modification of 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 (*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 actual 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.JsonResponse` and `moco_wrapper.util.response.ListingResponse` 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



## 8.1 Json Response

**class** `moco_wrapper.util.response.JsonResponse(response)`  
Class for handling http responses where the body is a single json 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 Listing Response

**class** `moco_wrapper.util.response.ListingResponse(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()
```

(continues on next page)

(continued from previous page)

```
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 already 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

## 8.3 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)
```

## 8.4 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

## 8.5 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

## 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 amount 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*) → 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)

**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*) → 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)

**Returns** The generated item

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

**See also:**

`generate_detail_position()` 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 position 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 position
- **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

## 9.3 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\_postion** (*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: int = 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** – Wether 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*) – 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
- **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()*

## A

ABSENCE (*moco\_wrapper.models.schedule.ScheduleAbsenceCode* attribute), 46

ACCEPTED (*moco\_wrapper.models.offer.OfferStatus* attribute), 31

Activity (class in *moco\_wrapper.models*), 13

ActivityRemoteService (class in *moco\_wrapper.models.activity*), 15

ANNUAL (*moco\_wrapper.models.project\_recurring\_expense.ProjectRecurringExpensePeriod* attribute), 40

archive() (*moco\_wrapper.models.Project* method), 32

ARCHIVED (*moco\_wrapper.models.offer.OfferStatus* attribute), 31

ASANA (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15

assigned() (*moco\_wrapper.models.Project* method), 32

auth (*moco\_wrapper.Moco* attribute), 11

authenticate() (*moco\_wrapper.Moco* method), 12

authenticate() (*moco\_wrapper.models.Session* method), 52

## B

BABY\_CARRIAGE (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

BASECAMP (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15

BASECAMP2 (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15

BASECAMP3 (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15

BELLS (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

BIANNUAL (*moco\_wrapper.models.project\_recurring\_expense.ProjectRecurringExpensePeriod* attribute), 40

BILLED (*moco\_wrapper.models.offer.OfferStatus* attribute), 31

BIWEEKLY (*moco\_wrapper.models.project\_recurring\_expense.ProjectRecurringExpensePeriod* attribute), 40

BUILDING (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

## C

CAR (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

class\_map (*moco\_wrapper.util.objector.DefaultObjector* attribute), 57

clear\_impersonation() (*moco\_wrapper.Moco* method), 12

COCKTAIL (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

Comment (class in *moco\_wrapper.models*), 16

CommentTargetType (class in *moco\_wrapper.models.comment*), 17

Company (class in *moco\_wrapper.models*), 18

CompanyType (class in *moco\_wrapper.models.company*), 20

Contact (class in *moco\_wrapper.models*), 20

CONTACT (*moco\_wrapper.models.comment.CommentTargetType* attribute), 17

ContactGender (class in *moco\_wrapper.models.contact*), 22

convert() (*moco\_wrapper.util.objector.DefaultObjector* method), 57

convert() (*moco\_wrapper.util.objector.NoErrorObjector* method), 59

convert() (*moco\_wrapper.util.objector.RawObjector* method), 60

create() (*moco\_wrapper.models.Activity* method), 13

create() (*moco\_wrapper.models.Comment* method), 16

create() (*moco\_wrapper.models.Company* method), 18

create() (*moco\_wrapper.models.ProjectRecurringExpensePeriod* method), 20

create() (*moco\_wrapper.models.Contact* method), 20

create() (*moco\_wrapper.models.Deal* method), 22

create() (*moco\_wrapper.models.DealCategory* method), 24

create() (*moco\_wrapper.models.Invoice* method), 25

[create\(\)](#) (*moco\_wrapper.models.InvoicePayment method*), 28  
[create\(\)](#) (*moco\_wrapper.models.Offer method*), 30  
[create\(\)](#) (*moco\_wrapper.models.Project method*), 32  
[create\(\)](#) (*moco\_wrapper.models.ProjectContract method*), 35  
[create\(\)](#) (*moco\_wrapper.models.ProjectExpense method*), 36  
[create\(\)](#) (*moco\_wrapper.models.ProjectPaymentSchedule method*), 41  
[create\(\)](#) (*moco\_wrapper.models.ProjectRecurringExpense method*), 39  
[create\(\)](#) (*moco\_wrapper.models.ProjectTask method*), 42  
[create\(\)](#) (*moco\_wrapper.models.Schedule method*), 44  
[create\(\)](#) (*moco\_wrapper.models.User method*), 47  
[create\(\)](#) (*moco\_wrapper.models.UserHoliday method*), 49  
[create\(\)](#) (*moco\_wrapper.models.UserPresence method*), 50  
[create\\_bulk\(\)](#) (*moco\_wrapper.models.Comment method*), 16  
[create\\_bulk\(\)](#) (*moco\_wrapper.models.InvoicePayment method*), 28  
[create\\_bulk\(\)](#) (*moco\_wrapper.models.ProjectExpense method*), 37  
[CREATED](#) (*moco\_wrapper.models.invoice.InvoiceStatus attribute*), 27  
[CREATED](#) (*moco\_wrapper.models.offer.OfferStatus attribute*), 31  
[current\\_page](#) (*moco\_wrapper.util.response.ListingResponse attribute*), 62  
[CUSTOMER](#) (*moco\_wrapper.models.comment.CommentTargetType attribute*), 17  
[CUSTOMER](#) (*moco\_wrapper.models.company.CompanyType attribute*), 20  
[CUSTOMER](#) (*moco\_wrapper.models.invoice.InvoiceChangeAddress attribute*), 28  
[CUSTOMER](#) (*moco\_wrapper.models.offer.OfferChangeAddress attribute*), 32  
[DEAL](#) (*moco\_wrapper.models.comment.CommentTargetType attribute*), 17  
[DealCategory](#) (*class in moco\_wrapper.models*), 24  
[DealStatus](#) (*class in moco\_wrapper.models.deal*), 23  
[DefaultObjector](#) (*class in moco\_wrapper.util.objector*), 57  
[DefaultRequestor](#) (*class in moco\_wrapper.util.requestor*), 53  
[delete\(\)](#) (*moco\_wrapper.models.Activity method*), 14  
[delete\(\)](#) (*moco\_wrapper.models.Comment method*), 16  
[delete\(\)](#) (*moco\_wrapper.models.DealCategory method*), 24  
[delete\(\)](#) (*moco\_wrapper.models.InvoicePayment method*), 29  
[delete\(\)](#) (*moco\_wrapper.models.ProjectContract method*), 35  
[delete\(\)](#) (*moco\_wrapper.models.ProjectExpense method*), 37  
[delete\(\)](#) (*moco\_wrapper.models.ProjectPaymentSchedule method*), 42  
[delete\(\)](#) (*moco\_wrapper.models.ProjectRecurringExpense method*), 39  
[delete\(\)](#) (*moco\_wrapper.models.ProjectTask method*), 43  
[delete\(\)](#) (*moco\_wrapper.models.Schedule method*), 44  
[delete\(\)](#) (*moco\_wrapper.models.User method*), 47  
[delete\(\)](#) (*moco\_wrapper.models.UserHoliday method*), 50  
[delete\(\)](#) (*moco\_wrapper.models.UserPresence method*), 51  
[disregard\(\)](#) (*moco\_wrapper.models.Activity method*), 14  
[disregard\(\)](#) (*moco\_wrapper.models.ProjectExpense method*), 37  
[DRAFT](#) (*moco\_wrapper.models.invoice.InvoiceStatus attribute*), 28  
[DROPPED](#) (*moco\_wrapper.models.deal.DealStatus attribute*), 24

## D

[data](#) (*moco\_wrapper.util.response.EmptyResponse attribute*), 64  
[data](#) (*moco\_wrapper.util.response.ErrorResponse attribute*), 64  
[data](#) (*moco\_wrapper.util.response.FileResponse attribute*), 63  
[data](#) (*moco\_wrapper.util.response.JsonResponse attribute*), 61  
[data](#) (*moco\_wrapper.util.response.ListingResponse attribute*), 62  
[Deal](#) (*class in moco\_wrapper.models*), 22

## E

[EmptyResponse](#) (*class in moco\_wrapper.util.response*), 64  
[error\\_class\\_map](#) (*moco\_wrapper.util.objector.DefaultObjector attribute*), 58  
[ErrorResponse](#) (*class in moco\_wrapper.util.response*), 64

## F

[FEMALE](#) (*moco\_wrapper.models.contact.ContactGender attribute*), 22  
[file](#) (*moco\_wrapper.util.response.FileResponse attribute*), 63

FileResponse (class in *moco\_wrapper.util.response*), 63

full\_domain (*moco\_wrapper.Moco* attribute), 12

## G

generate () (*moco\_wrapper.util.generator.InvoicePaymentGenerator* method), 65

generate () (*moco\_wrapper.util.generator.ProjectExpenseGenerator* method), 69

generate\_description () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 66

generate\_description () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 67

generate\_detail\_postion () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 66

generate\_detail\_postion () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 68

generate\_item () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 66

generate\_item () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 68

generate\_lump\_position () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 67

generate\_lump\_position () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 68

generate\_pagebreak () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 67

generate\_pagebreak () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 69

generate\_separator () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 67

generate\_separator () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 69

generate\_subtotal () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 67

generate\_subtotal () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 69

generate\_title () (*moco\_wrapper.util.generator.InvoiceItemGenerator* method), 67

generate\_title () (*moco\_wrapper.util.generator.OfferItemGenerator* method), 69

get () (*moco\_wrapper.models.Activity* method), 14

get () (*moco\_wrapper.models.Comment* method), 16

get () (*moco\_wrapper.models.Company* method), 19

get () (*moco\_wrapper.models.Contact* method), 21

get () (*moco\_wrapper.models.Deal* method), 23

get () (*moco\_wrapper.models.DealCategory* method), 25

get () (*moco\_wrapper.models.Invoice* method), 26

get () (*moco\_wrapper.models.InvoicePayment* method), 29

get () (*moco\_wrapper.models.Offer* method), 30

get () (*moco\_wrapper.models.Project* method), 33

get () (*moco\_wrapper.models.ProjectContract* method), 35

get () (*moco\_wrapper.models.ProjectExpense* method), 37

get () (*moco\_wrapper.models.ProjectPaymentSchedule* method), 42

get () (*moco\_wrapper.models.ProjectRecurringExpense* method), 39

get () (*moco\_wrapper.models.ProjectTask* method), 43

get () (*moco\_wrapper.models.Schedule* method), 44

get () (*moco\_wrapper.models.Unit* method), 46

get () (*moco\_wrapper.models.User* method), 47

get () (*moco\_wrapper.models.UserEmployment* method), 48

get () (*moco\_wrapper.models.UserHoliday* method), 50

get () (*moco\_wrapper.models.UserPresence* method), 51

get\_class\_name\_from\_request\_url () (*moco\_wrapper.util.objector.DefaultObjector* method), 58

get\_error\_class\_name\_from\_response\_status\_code () (*moco\_wrapper.util.objector.DefaultObjector* method), 59

getall () (*moco\_wrapper.models.ProjectExpense* method), 37

getlist () (*moco\_wrapper.models.Activity* method), 14

getlist () (*moco\_wrapper.models.Comment* method), 17

getlist () (*moco\_wrapper.models.Company* method), 19

getlist () (*moco\_wrapper.models.Contact* method), 21

getlist () (*moco\_wrapper.models.Deal* method), 23

getlist () (*moco\_wrapper.models.DealCategory* method), 25

getlist () (*moco\_wrapper.models.Invoice* method), 26

getlist () (*moco\_wrapper.models.InvoicePayment* method), 29

getlist () (*moco\_wrapper.models.Offer* method), 30

getlist () (*moco\_wrapper.models.Project* method), 33

[getlist\(\)](#) (*moco\_wrapper.models.ProjectContract* method), 35  
[getlist\(\)](#) (*moco\_wrapper.models.ProjectExpense* method), 38  
[getlist\(\)](#) (*moco\_wrapper.models.ProjectPaymentSchedule* method), 42  
[getlist\(\)](#) (*moco\_wrapper.models.ProjectRecurringExpense* method), 40  
[getlist\(\)](#) (*moco\_wrapper.models.ProjectTask* method), 43  
[getlist\(\)](#) (*moco\_wrapper.models.Schedule* method), 44  
[getlist\(\)](#) (*moco\_wrapper.models.Unit* method), 46  
[getlist\(\)](#) (*moco\_wrapper.models.User* method), 48  
[getlist\(\)](#) (*moco\_wrapper.models.UserEmployment* method), 49  
[getlist\(\)](#) (*moco\_wrapper.models.UserHoliday* method), 50  
[getlist\(\)](#) (*moco\_wrapper.models.UserPresence* method), 51  
[GITHUB](#) (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15  
[GRADUATION\\_CAP](#) (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

## H

[headers](#) (*moco\_wrapper.Moco* attribute), 12  
[HOLIDAY](#) (*moco\_wrapper.models.schedule.ScheduleAbsenceCode* attribute), 46  
[HOME](#) (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

## I

[IGNORED](#) (*moco\_wrapper.models.invoice.InvoiceStatus* attribute), 28  
[impersonate\(\)](#) (*moco\_wrapper.Moco* method), 12  
[INFO\\_CIRCLE](#) (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46  
[Invoice](#) (class in *moco\_wrapper.models*), 25  
[INVOICE](#) (*moco\_wrapper.models.comment.CommentTargetType* attribute), 17  
[INVOICE](#) (*moco\_wrapper.models.invoice.InvoiceChangeAddress* attribute), 28  
[InvoiceChangeAddress](#) (class in *moco\_wrapper.models.invoice*), 28  
[InvoiceItemGenerator](#) (class in *moco\_wrapper.util.generator*), 66  
[InvoicePayment](#) (class in *moco\_wrapper.models*), 28  
[InvoicePaymentGenerator](#) (class in *moco\_wrapper.util.generator*), 65  
[InvoiceStatus](#) (class in *moco\_wrapper.models.invoice*), 27  
[is\\_last](#) (*moco\_wrapper.util.response.ListingResponse* attribute), 62

## J

[is\\_recoverable](#) (*moco\_wrapper.util.response.ErrorResponse* attribute), 64  
[items](#) (*moco\_wrapper.util.response.ListingResponse* attribute), 62

## L

[last\\_page](#) (*moco\_wrapper.util.response.ListingResponse* attribute), 62  
[ListingResponse](#) (class in *moco\_wrapper.util.response*), 61  
[locked\(\)](#) (*moco\_wrapper.models.Invoice* method), 26  
[LOST](#) (*moco\_wrapper.models.deal.DealStatus* attribute), 24

## M

[MALE](#) (*moco\_wrapper.models.contact.ContactGender* attribute), 22  
[MITE](#) (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15  
[Moco](#) (class in *moco\_wrapper*), 11  
[MONTHLY](#) (*moco\_wrapper.models.project\_recurring\_expense.ProjectRecurringExpense* attribute), 41  
[MOON](#) (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

## N

[next\\_page](#) (*moco\_wrapper.util.response.ListingResponse* attribute), 62  
[NoErrorObjector](#) (class in *moco\_wrapper.util.objector*), 59  
[NoRetryRequestor](#) (class in *moco\_wrapper.util.requestor*), 54

## O

[objector](#) (*moco\_wrapper.Moco* attribute), 12  
[Offer](#) (class in *moco\_wrapper.models*), 30  
[OFFER](#) (*moco\_wrapper.models.comment.CommentTargetType* attribute), 17  
[OFFER](#) (*moco\_wrapper.models.offer.OfferChangeAddress* attribute), 32  
[OfferChangeAddress](#) (class in *moco\_wrapper.models.offer*), 31  
[OFFERCONFIRMATION](#) (*moco\_wrapper.models.comment.CommentTargetType* attribute), 17  
[OfferItemGenerator](#) (class in *moco\_wrapper.util.generator*), 67



- OfferStatus (class in moco\_wrapper.models.offer), 31
- ORGANIZATION (moco\_wrapper.models.company.CompanyType attribute), 20
- OVERDUE (moco\_wrapper.models.invoice.InvoiceStatus attribute), 28
- ## P
- page\_size (moco\_wrapper.util.response.ListingResponse attribute), 63
- PAID (moco\_wrapper.models.invoice.InvoiceStatus attribute), 28
- PARTIALLY\_BILLED (moco\_wrapper.models.offer.OfferStatus attribute), 31
- PARTIALLY\_PAID (moco\_wrapper.models.invoice.InvoiceStatus attribute), 28
- pdf () (moco\_wrapper.models.Invoice method), 27
- pdf () (moco\_wrapper.models.Offer method), 31
- PENDING (moco\_wrapper.models.deal.DealStatus attribute), 24
- POTENTIAL (moco\_wrapper.models.deal.DealStatus attribute), 24
- Project (class in moco\_wrapper.models), 32
- PROJECT (moco\_wrapper.models.comment.CommentTargetType attribute), 17
- PROJECT (moco\_wrapper.models.invoice.InvoiceChangeAddress attribute), 28
- PROJECT (moco\_wrapper.models.project.ProjectBillingVariant attribute), 34
- ProjectBillingVariant (class in moco\_wrapper.models.project), 34
- ProjectContract (class in moco\_wrapper.models), 35
- ProjectExpense (class in moco\_wrapper.models), 36
- ProjectExpenseGenerator (class in moco\_wrapper.util.generator), 69
- ProjectPaymentSchedule (class in moco\_wrapper.models), 41
- ProjectRecurringExpense (class in moco\_wrapper.models), 39
- ProjectRecurringExpensePeriod (class in moco\_wrapper.models.project\_recurring\_expense), 40
- ProjectTask (class in moco\_wrapper.models), 42
- PUBLIC\_HOLIDAY (moco\_wrapper.models.schedule.ScheduleAbsenceCode attribute), 46
- RawRequestor (class in moco\_wrapper.util.requestor), 54
- request () (moco\_wrapper.models.Project method), 33
- request () (moco\_wrapper.Moco method), 12
- request () (moco\_wrapper.util.requestor.DefaultRequestor method), 53
- request () (moco\_wrapper.util.requestor.NoRetryRequestor method), 54
- request () (moco\_wrapper.util.requestor.RawRequestor method), 54
- requestor (moco\_wrapper.Moco attribute), 13
- response (moco\_wrapper.util.response.EmptyResponse attribute), 64
- response (moco\_wrapper.util.response.ErrorResponse attribute), 64
- response (moco\_wrapper.util.response.FileResponse attribute), 63
- response (moco\_wrapper.util.response.JsonResponse attribute), 61
- response (moco\_wrapper.util.response.ListingResponse attribute), 63
- ## S
- Schedule (class in moco\_wrapper.models), 44
- ScheduleAbsenceCode (class in moco\_wrapper.models.schedule), 45
- ScheduleSymbol (class in moco\_wrapper.models.schedule), 46
- SENT (moco\_wrapper.models.invoice.InvoiceStatus attribute), 28
- SENT (moco\_wrapper.models.offer.OfferStatus attribute), 31
- Session (class in moco\_wrapper.models), 52
- session (moco\_wrapper.Moco attribute), 13
- session (moco\_wrapper.util.requestor.DefaultRequestor attribute), 53
- session (moco\_wrapper.util.requestor.NoRetryRequestor attribute), 54
- SICK\_DAY (moco\_wrapper.models.schedule.ScheduleAbsenceCode attribute), 46
- start\_timer () (moco\_wrapper.models.Activity method), 14
- stop\_timer () (moco\_wrapper.models.Activity method), 14
- sub\_absence\_code (moco\_wrapper.models.company.CompanyType attribute), 20
- ## T
- timesheet () (moco\_wrapper.models.Invoice method), 27
- TOGGL (moco\_wrapper.models.activity.ActivityRemoteService attribute), 15
- ## Q
- QUARTERLY (moco\_wrapper.models.project\_recurring\_expense.ProjectRecurringExpensePeriod attribute), 41
- ## R
- RawObjector (class in moco\_wrapper.util.objector), 60

total (*moco\_wrapper.util.response.ListingResponse* attribute), 63

touch() (*moco\_wrapper.models.UserPresence* method), 51

TRELLO (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 15

## U

unarchive() (*moco\_wrapper.models.Project* method), 34

UNDEFINED (*moco\_wrapper.models.contact.ContactGender* attribute), 22

Unit (class in *moco\_wrapper.models*), 46

UNPLANNED (*moco\_wrapper.models.schedule.ScheduleAbsence* attribute), 46

update() (*moco\_wrapper.models.Activity* method), 15

update() (*moco\_wrapper.models.Comment* method), 17

update() (*moco\_wrapper.models.Company* method), 19

update() (*moco\_wrapper.models.Contact* method), 21

update() (*moco\_wrapper.models.Deal* method), 23

update() (*moco\_wrapper.models.DealCategory* method), 25

update() (*moco\_wrapper.models.InvoicePayment* method), 29

update() (*moco\_wrapper.models.Project* method), 34

update() (*moco\_wrapper.models.ProjectContract* method), 36

update() (*moco\_wrapper.models.ProjectExpense* method), 38

update() (*moco\_wrapper.models.ProjectPaymentSchedule* method), 42

update() (*moco\_wrapper.models.ProjectRecurringExpense* method), 40

update() (*moco\_wrapper.models.ProjectTask* method), 43

update() (*moco\_wrapper.models.Schedule* method), 45

update() (*moco\_wrapper.models.User* method), 48

update() (*moco\_wrapper.models.UserHoliday* method), 50

update() (*moco\_wrapper.models.UserPresence* method), 51

update\_status() (*moco\_wrapper.models.Invoice* method), 27

User (class in *moco\_wrapper.models*), 47

USER (*moco\_wrapper.models.comment.CommentTargetType* attribute), 18

USER (*moco\_wrapper.models.project.ProjectBillingVariant* attribute), 35

UserEmployment (class in *moco\_wrapper.models*), 48

UserHoliday (class in *moco\_wrapper.models*), 49

UserPresence (class in *moco\_wrapper.models*), 50

USERS (*moco\_wrapper.models.schedule.ScheduleSymbol* attribute), 46

## W

WEEKLY (*moco\_wrapper.models.project\_recurring\_expense.ProjectRecurringExpense* attribute), 41

WON (*moco\_wrapper.models.deal.DealStatus* attribute), 24

write\_to\_file() (*moco\_wrapper.util.response.FileResponse* method), 63

UNDERLIST (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 16

## Y

YOUTRACK (*moco\_wrapper.models.activity.ActivityRemoteService* attribute), 16