
Zammad API Client Documentation

Release 3.0.0

Joe Paul

Jun 27, 2023

Contents

1	Zammad API Client	3
1.1	Quickstart	3
1.2	General Methods	4
1.3	Additional Resource Methods	4
1.4	Contributing	4
1.5	Credits	5
2	Installation	7
2.1	Stable release	7
2.2	From sources	7
3	Usage	9
3.1	General Methods	9
3.2	User Resource	11
3.3	Ticket Resource	11
3.4	Link Resource	11
3.5	TicketArticleAttachment Resource	12
3.6	Object Resource	12
3.7	Using “On behalf of”	12
4	Contributing	13
4.1	Types of Contributions	13
4.2	Get Started!	14
4.3	Pull Request Guidelines	15
4.4	Tips	15
5	Credits	17
5.1	Development Lead	17
5.2	Contributors	17
6	History	19
6.1	3.0.0 (2023-02-27)	19
6.2	2.0.0 (2023-02-14)	19
6.3	1.1.0 (2022-08-04)	19
6.4	1.0.1 (2021-05-28)	19
6.5	0.1.7 (2019-04-25)	19
6.6	0.1.6 (2019-04-11)	20

6.7	0.1.5 (2018-05-11)	20
6.8	0.1.4 (2018-01-02)	20
6.9	0.1.3 (2017-11-13)	20
6.10	0.1.2 (2017-10-26)	20
6.11	0.1.0 (2017-10-25)	20
7	Indices and tables	21

Contents:

CHAPTER 1

Zammad API Client

Python API client for zammad

- Free software: MIT license
- Documentation: <https://zammad-py.readthedocs.io>.

1.1 Quickstart

```
from zammad_py import ZammadAPI

# Initialize the client with the URL, username, and password
# Note the Host URL should be in this format: 'https://zammad.example.org/api/v1/'
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')

# Example: Access all users
this_page = client.user.all()
for user in this_page:
    print(user)

# Example: Get information about the current user
print(client.user.me())

# Example: Create a ticket
params = {
    "title": "Help me!",
    "group": "2nd Level",
    "customer": "david@example.com",
    "article": {
```

(continues on next page)

(continued from previous page)

```
"subject": "My subject",
"body": "I am a message!",
"type": "note",
"internal": false
}
}
new_ticket = client.ticket.create(params=params)
```

1.2 General Methods

Most resources support these methods:

.all(): Returns a paginated response with the current page number and a list of elements.

.next_page(): Returns the next page of the current pagination object.

.prev_page(): Returns the previous page of the current pagination object.

.search(params): Returns a paginated response based on the search parameters.

.find(id): Returns a single object with the specified ID.

.create(params): Creates a new object with the specified parameters.

.update(params): Updates an existing object with the specified parameters.

.destroy(id): Deletes an object with the specified ID.

1.3 Additional Resource Methods

User resource also has the *.me()* method to get information about the current user.

Ticket resource also has the *.articles()* method to get the articles associated with a ticket.

Link resource has methods to list, add, and delete links between objects.

TicketArticleAttachment resource has the *.download()* method to download a ticket attachment.

Object resource has the *.execute_migrations()* method to run migrations on an object.

You can set the *on_behalf_of* attribute of the ZammadAPI instance to do actions on behalf of another user.

1.4 Contributing

The Zammad API Client (zammad_py) welcomes contributions.

You can contribute by reporting bugs, fixing bugs, implementing new features, writing documentation, and submitting feedback.

To get started, see the contributing section in the docs!

Please ensure that your changes include tests and updated documentation if necessary.

1.5 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

2.1 Stable release

To install Zammad API Client, run this command in your terminal:

```
$ pip install zammad_py
```

This is the preferred method to install Zammad API Client, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

2.2 From sources

The sources for Zammad API Client can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/joeirimpan/zammad_py
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/joeirimpan/zammad_py/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```


CHAPTER 3

Usage

To use Zammad API Client in a project:

```
from zammad_py import ZammadAPI
# Note the Host URL should be in this format: 'https://zammad.example.org/api/v1/'
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
```

Zammad Resources are implemented as an abstract class (Resource), meaning most objects have the same mechanisms. Because Zammad uses pagination (<https://docs.zammad.org/en/latest/api/intro.html#pagination>), the pagination is very helpful.

3.1 General Methods

Most Resources support these methods.

.all()

Returns a paginated response with `_page` containing the page number and `_items` containing a list of elements. This allows to iterate over the objects.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
# all return a paginated response
this_page = client.user.all()
# Iterate through page object
for user in this_page:
    print(user)
```

.next_page()

Can be called from the current page (returned by `.all()`)
Contains the next page object if there are any.

```
next_page = this_page.next_page()
```

.prev_page()

Can be called from the current page (pagination object)

Contains the previous page object if there are any.

```
prev_page = this_page.prev_page()
```

.search(search_string)

Searches the object with a Zammad search query

Learn more about Zammad Search queries here: <https://user-docs.zammad.org/en/latest/advanced/search.html>

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
client.ticket.search('Search Content')
```

.find(id)

Displays a Resource if you know the id. (Returns a dict)

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
client.ticket.find(<TICKETID>)
```

.create(params)

Creates a new Resource.

You can find the required structure for the params in the Zammad API Documentation.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
params = {
    "title": "Help me!",
    "group": "2nd Level",
    "customer": "david@example.com",
    "article": {
        "subject": "My subject",
        "body": "I am a message!",
        "type": "note",
        "internal": false
    }
}
new_ticket = client.ticket.create(params=params)
```

.update(params)

Updates a resource.

You can find the required structure for the params in the Zammad API Documentation, or use a dict provided by the .find(id) functionality.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
org = client.organization.find(<ID>)
org['name'] = 'NewCompanyName Ltd.'
client.organization.update(id=org['id'], params=org)
```

.destroy(id)

Deletes a Resource.

For some resources, you need special privileges. Refer to the Zammad API Documentation.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
client.organization.destroy(<ID>)
```

Available Resources user organization group ticket link ticketarticle ticketarticleplain ticketpriority ticketstate object taglist

3.2 User Resource

The User resource also has the `me()` method to get information about the current user.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
print(client.user.me())
```

3.3 Ticket Resource

The Ticket resource also has the `articles()` method to get the articles associated to the ticket.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
print(client.ticket.find(<ID>))
ticketarticles = client.ticket.articles
print(ticketarticles)
```

Further, it has the `merge()` method, that allows to merge two tickets. (This is not documented in the Zammad API Documentation) The method requires the Ticket id of the Child (The ticket you want to merge into the parent) and the Ticket Number of the Parent Ticket. (The ticket you want to contain the articles of the child after merging.)

Important: If you want to use the merge functionality, you need to use password, not `http_token` for your authentication.

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
client.ticket.merge(id=<ID>, number=<NUMBER>)
```

3.4 Link Resource

The Link resource also has methods to list, add and delete Links between objects.

zammad_py.api.Link.get()

This returns all links associated with the ticket ID provided

zammad_py.api.Link.add()

Create a Link between two objects. (Currently, and by default Tickets)

zammad_py.api.Link.remove()

Remove a Link between two objects. (Currently, and by default Tickets)

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
print(client.link.get(<ID>))
ticketarticles = client.ticket.articles
print(ticketarticles)
```

3.5 TicketArticleAttachment Resource

The TicketArticleAttachment resource has the `download()` method.

```
"""Download the ticket attachment associated with the ticket id

:param id: Ticket attachment id
:param article_id: Ticket article id
:param ticket_id: Ticket id
"""
```

3.6 Object Resource

The Object resource has the `execute_migrations()` method to run the migrations of an object.

3.7 Using “On behalf of”

To do actions on behalf of another user, just set the `on_behalf_of` attribute on the instance of ZammadAPI

```
from zammad_py import ZammadAPI
client = ZammadAPI(url='<HOST>', username='<USERNAME>', password='<PASSWORD>')
client.on_behalf_of = 'test@user.com'
# Do stuff...
```

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/joeirimpan/zammad_py/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

4.1.4 Write Documentation

The Zammad API Client could always use more documentation, whether as part of the official Zammad API Client docs, in docstrings, or even on the web in blog posts, articles, and such.

To help with documentation, look in the *docs* folder. If you want to generate the documentation locally to see how it looks, you need to install sphinx

1. **Assuming you installed the dependencies with *poetry install*, you can run to generate the html files.** `$ python3 -m sphinx -T -E -b html -d docs/_build/doctrees -D language=en docs/ ./_readthedocs/html`
2. Navigate to the new `_readthedocs/html` folder and open `index.html` with a Browser to see a preview of the documentation.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/joeirimpan/zammad_py/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *zammad_py* for local development.

1. Install poetry

```
$ pip3 install poetry flask8 sphinx
```

2. Fork the *zammad_py* repo on GitHub.
3. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/zammad_py.git
```

4. This project uses Poetry (<https://python-poetry.org/>). Set up the projects dependencies like this:

```
$ pip3 install poetry $ cd zammad_py/ $ poetry install
```

Poetry will create a virtual environment under your home directory. You can see if that worked by running:

```
$ ls ~/.cache/pypoetry/virtualenvs/zammad-py*
```

Then activate the environment like this:

```
$ ~/.cache/pypoetry/virtualenvs/zammad-py*/bin/activate
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 zammad_py tests
$ py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python $\geq 3.8.1$

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_zammad_py
```


5.1 Development Lead

- Joe Paul <joeirimpan@gmail.com>

5.2 Contributors

Matthew Frost <matthew@uptic.nl>

6.1 3.0.0 (2023-02-27)

- release/3.0.0 Add Pagination to search. This is an incompatible change to search API in olders versions.

6.2 2.0.0 (2023-02-14)

- release/2.0.0 Documentation rewrite with more examples for use Rewrote Contributing Guide Added new Resource: Link Fixed Tests, Create and delete functions to not cause errors by calling the urls with expand=true added.

6.3 1.1.0 (2022-08-04)

- release/1.1.0 Small Bugfixes, Linting Added TicketArticlePlain

6.4 1.0.1 (2021-05-28)

- release/1.0.1 feat: Take url instead of host Update pytest, requests Fix Documentation

6.5 0.1.7 (2019-04-25)

- release/0.1.7 feat: Make username, pass optional

6.6 0.1.6 (2019-04-11)

- release/0.1.6 Update vcrpy, requests Fix Pagination, Add per_page getter / setter fix: Add context manager for doing req with behalf header

6.7 0.1.5 (2018-05-11)

- release/0.1.5 Add more python versions to the test matrix

6.8 0.1.4 (2018-01-02)

- release/0.1.4: Released version 0.1.4 Support indexing on paginated response Update pytest from 3.2.5 to 3.3.0 Close session at exit Update pytest from 3.2.4 to 3.2.5 Update pytest from 3.2.3 to 3.2.4 API improvements Update README.rst Add pagination tests

6.9 0.1.3 (2017-11-13)

- release/0.1.3: Released version 0.1.3 Add pagination support Add more resource endpoints Pin vcrpy to latest version 1.11.1 Pin pytest to latest version 3.2.3 Pin flake8 to latest version 3.5.0 Pin requests to latest version 2.18.4

6.10 0.1.2 (2017-10-26)

- release/0.1.2: Released version 0.1.2 Add toggle flag to use SSL or not Add tests for group resource Update README.md Add tests for ticket resource Fix json decode error
master

6.11 0.1.0 (2017-10-25)

- Released version 0.1.1 Test for all CRUD ops Add update method on base resource Add tests Update method in API Flake8ify Add basic zammad api Initial Boilerplate

CHAPTER 7

Indices and tables

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