# Pyzotero Documentation

***Release 1.2.13***

**Stephan Hügel**

**Jun 22, 2017**

# Contents


**1 Getting started (short version) 3**

**2 Installation, testing, usage (longer version) 5**
- 2.1 Installation . . . 5
- 2.2 Installing development versions . . . 5
- 2.3 Testing . . . 5
- 2.4 Building Documentation . . . 6
- 2.5 Reporting issues . . . 6
- 2.6 General Usage . . . 6

**3 Read API Methods 7**
- 3.1 Retrieving Items . . . 7
- 3.2 Retrieving Files . . . 9
- 3.3 Retrieving Collections . . . 10
- 3.4 Retrieving groups . . . 11
- 3.5 Retrieving Tags . . . 12
- 3.6 Retrieving Version Information . . . 12
- 3.7 Full–Text Content . . . 12
- 3.8 The `follow()`, and `everything()` methods . . . 13
- 3.9 Retrieving item counts . . . 15
- 3.10 Retrieving last modified version . . . 15
- 3.11 Search / Request Parameters for Read API calls . . . 15

**4 Write API Methods 19**
- 4.1 Item Methods . . . 19
- 4.2 Adding tags . . . 22
- 4.3 Collection Methods . . . 23

**5 Notes 25**

**6 License 27**

**7 Cat Picture 29**

**Python Module Index 31**

Pyzotero is a Python wrapper for the Zotero API (v3).

# CHAPTER 1

## Getting started (short version)

1. In a shell / prompt: `pip install pyzotero`
2. You'll need the ID of the personal or group library you want to access:

- Your **personal library ID** is available here, in the section `Your userID for use in API calls`
- For **group libraries**, the ID can be found by opening the group's page: `https://www.zotero.org/groups/groupname`, and hovering over the `group settings` link. The ID is the integer after `/groups/`

3. You'll also need*[0] to get an **API key** from the Zotero site
4. Are you accessing your own Zotero library? `library_type` is `user`
5. Are you accessing a shared group library? `library_type` is `group`

```
    from pyzotero import zotero
    zot = zotero.Zotero(library_id, library_type, api_key)
    items = zot.top(limit=5)
    # we've retrieved the latest five top-level items in our library
    # we can print each item's item type and ID
    for item in items:
    print('Item Type: %s | Key: %s' % (item['data']['itemType'], item['data
↪']['key']))
```

Refer to the *Read API Methods* and *Write API Methods*.

[0] This isn't strictly true: you only need an API key for personal libraries and non-public group libraries.

CHAPTER 2

# Installation, testing, usage (longer version)

## Installation

Using pip: `pip install pyzotero` As of v1.0.0, Pyzotero is also available as a wheel.

From a local clone, if you wish to install Pyzotero from a specific branch:

```
git clone git://github.com/urschrei/pyzotero.git
cd pyzotero
git checkout dev
pip install .
```

The Pyzotero source tarball is also available from PyPI

The feedparser (>= 0.5.1), pytz , and Requests libraries are required. They will be automatically installed when installing Pyzotero using pip. For versions of Python below 2.7, the ordereddict module is also required. This optional dependency can be included with Pyzotero with the command `pip install pyzotero[ordereddict]`.

## Installing development versions

Pyzotero remains in development as of February 2015. Unstable development versions can be found on the Github dev branch, and installed directly from there using pip: `pip install -e git+https://github.com/urschrei/pyzotero.git@dev#egg=pyzotero`, or from the checked-out `dev` branch on a local clone, as in the example above.

## Testing

Testing requires the `Nose`, `HTTPretty`, and `Python-Dateutil` packages.

Run `test_zotero.py` in the `pyzotero/test` directory, or, using Nose, `nosetests` from the top-level directory. If you wish to see coverage statistics, run `nosetests --with-coverage --cover-package=pyzotero`.

## Building Documentation

If you wish to build Pyzotero's documentation for offline use, it can be built from the `doc` directory of a local git repo by running `make` followed by the desired output format(s) (`html`, `epub`, `latexpdf` etc.)

This functionality requires Sphinx. See the Sphinx documentation for full details.

## Reporting issues

If you encounter an error while using Pyzotero, please open an issue on its Github issues page.

## General Usage

---

**Important:** A given `Zotero` instance is bound to the library or group used to create it. Thus, if you create a `Zotero` instance with a `library_id` of `67` and a `library_type` of `group`, its item methods will only operate upon that group. Similarly, if you create a `Zotero` instance with your own `library_id` and a `library_type` of `user`, the instance will be bound to your Zotero library.

---

First, create a new Zotero instance:

**class** `zotero.`**`Zotero`**(*library_id*, *library_type*[, *api_key*, *preserve_json_order* ])

**Parameters**

- **`library_id`** (*`str`*) – a valid Zotero API user ID
- **`library_type`** (*`str`*) – a valid Zotero API library type: **user** or **group**
- **`api_key`** (*`str`*) – a valid Zotero API user key
- **`preserve_json_order`** (*`bool`*) – Load JSON returns with OrderedDict to preserve their order

Example:

```
from pyzotero import zotero
zot = zotero.Zotero('123', 'user', 'ABC1234XYZ')
# we now have a Zotero object, zot, and access to all its methods
first_ten = zot.items(limit=10)
# a list containing dicts of the ten most recently modified library items
```

# CHAPTER 3

# Read API Methods

**Note:** All search/request parameters inside square brackets are **optional**. Methods such as *Zotero.top()*, *Zotero.items()* etc. can be called with no additional parameters if you wish.

**Tip:** The Read API returns 25 results by default (the API documentation claims 50). In the interests of usability, Pyzotero returns 100 items by default, by setting the API `limit` parameter to 100, unless it's set by the user. If you wish to retrieve e.g. all top-level items without specifiying a `limit` parameter, you'll have to wrap your call with *Zotero.everything()*: `results = zot.everything(zot.top())`.

`Zotero.`**`key_info`**`()`

Returns info about the user and group library permissions associated with the current `Zotero` instance, based on the API key. Together with *Zotero.groups()*, this allows all accessible resources to be determined.

**Return type** dict

## Retrieving Items

**Tip:** In contrast to the v1 API, a great deal of additional metadata is now returned. In most cases, simply accessing items by referring to their `item['data']` key will suffice.

The following methods will retrieve either user or group items, depending on the value (`user` or `group`) used to create the `Zotero` instance:

`Zotero.`**`items`**`(`[*search/request parameters*]`)`

Returns Zotero library items

**Return type** list of dicts

Zotero.**top**([*search/request parameters*])
Returns top-level Zotero library items

**Return type** list of dicts

Zotero.**trash**([*search/request parameters*])
Returns library items from the library's trash

**Return type** list of dicts

Zotero.**deleted**([*search/request parameters*])
Returns deleted collections, library items, tags, searches and settings (requires "since=" parameter)

**Return type** list of dicts

Zotero.**item**(*itemID*[, *search/request parameters*])
Returns a specific item

**Parameters** **itemID** (*str*) – a zotero item ID

**Return type** list of dicts

Zotero.**children**(*itemID*[, *search/request parameters*])
Returns the child items of a specific item

**Parameters** **itemID** (*str*) – a zotero item ID

**Return type** list of dicts

Zotero.**collection_items**(*collectionID*[, *search/request parameters*])
Returns items from the specified collection. This includes sub-collection items

**Parameters** **collectionID** (*str*) – a Zotero collection ID

**Return type** list of dicts

Zotero.**collection_items_top**(*collectionID*[, *search/request parameters*])
Returns top-level items from the specified collection.

**Parameters** **collectionID** (*str*) – a Zotero collection ID

**Return type** list of dicts

Zotero.**get_subset**(*itemIDs*[, *search/request parameters*])
Retrieve an arbitrary set of non-adjacent items. Limited to 50 items per call.

**Parameters** **itemIDs** (*list*) – a list of Zotero Item IDs

**Return type** list of dicts

Example of returned item data:

```
[{u'data': {u'ISBN': u'0810116820',
            u'abstractNote': u'',
            u'accessDate': u'',
            u'archive': u'',
            u'archiveLocation': u'',
            u'callNumber': u'HIB 828.912 BEC:3g N9',
            u'collections': [u'2UNGXMU9'],
            u'creators': [{u'creatorType': u'author',
                           u'firstName': u'Daniel',
                           u'lastName': u'Katz'}],
            u'date': u'1999',
            u'dateAdded': u'2010-01-04T14:50:40Z',
            u'dateModified': u'2014-08-06T11:28:41Z',
            u'edition': u'',
```

```
          u'extra': u'',
          u'itemType': u'book',
          u'key': u'VDNIEAPH',
          u'language': u'',
          u'libraryCatalog': u'library.catalogue.tcd.ie Library Catalog',
          u'numPages': u'',
          u'numberOfVolumes': u'',
          u'place': u'Evanston, Ill',
          u'publisher': u'Northwestern University Press',
          u'relations': {u'dc:replaces': u'http://zotero.org/users/436/
↪items/9TXN8QUD'},
          u'rights': u'',
          u'series': u'',
          u'seriesNumber': u'',
          u'shortTitle': u'Saying I No More',
          u'tags': [{u'tag': u'Beckett, Samuel', u'type': 1},
                    {u'tag': u'Consciousness in literature', u'type': 1},
                    {u'tag': u'English prose literature', u'type': 1},
                    {u'tag': u'Ireland', u'type': 1},
                    {u'tag': u'Irish authors', u'type': 1},
                    {u'tag': u'Modernism (Literature)', u'type': 1},
                    {u'tag': u'Prose', u'type': 1},
                    {u'tag': u'Self in literature', u'type': 1},
                    {u'tag': u'Subjectivity in literature', u'type': 1}],
          u'title': u'Saying I No More: Subjectivity and Consciousness in␣
↪The Prose of Samuel Beckett',
          u'url': u'',
          u'version': 792,
          u'volume': u''},
 u'key': u'VDNIEAPH',
 u'library': {u'id': 436,
              u'links': {u'alternate': {u'href': u'https://www.zotero.org/
↪urschrei',
                                        u'type': u'text/html'}},
              u'name': u'urschrei',
              u'type': u'user'},
 u'links': {u'alternate': {u'href': u'https://www.zotero.org/urschrei/items/
↪VDNIEAPH',
                            u'type': u'text/html'},
            u'self': {u'href': u'https://api.zotero.org/users/436/items/
↪VDNIEAPH',
                      u'type': u'application/json'}},
 u'meta': {u'creatorSummary': u'Katz',
           u'numChildren': 0,
           u'parsedDate': u'1999-00-00'},
 u'version': 792}]
```

See *'Hello World'* example, above

## Retrieving Files

`Zotero.`**`file`**(*itemID*[, *search/request parameters* ])

Returns the raw file content of an item. This can be dumped like so:

```
with open('article.pdf', 'wb') as f:
  f.write(zot.file('BM8MZJBB'))
```

**Parameters** **itemID** (*str*) – a zotero item ID

**Return type** binary string

Zotero.**dump**(*itemID*[, *filename*, *path*])
A convenient wrapper around *Zotero.file()*. Writes an attachment to disk using the optional path and filename. If neither are supplied, the file is written to the current working directory, and a *Zotero.item()* call is first made to determine the attachment filename. No error checking is done regarding the path.

**Note:** HTML snapshots will be dumped as zip files, with "zip" appended to the file name.

```
# write a file to the current working directory using the stored filename
zot.dump('BM8MZJBB')
# write the same file to a different path, with a new name
zot.dump('BM8MZJBB', 'article_1.pdf', '/home/beckett/pdfs')
```

**Parameters**

- **itemID** (*str*) – a zotero item ID
- **filename** (*str*) – (optional) an alternate filename
- **path** (*str*) – (optional) a valid path for the file

**Return type** None

File retrieval and dumping should work for most common document, audio and video file formats. If you encounter an error, please open an issue.

## Retrieving Collections

Zotero.**collections**([*search/request parameters*])
Returns a library's collections. **This includes subcollections**.

**Return type** list of dicts

Zotero.**collections_top**([*search/request parameters*])
Returns a library's top-level collections.

**Return type** list of dicts

Zotero.**collection**(*collectionID*[, *search/request parameters*])
Returns a specific collection

**Parameters** **collectionID** (*str*) – a Zotero library collection ID

**Return type** dict

Zotero.**collections_sub**(*collectionID*[, *search/request parameters*])
Returns the sub-collections of a specific collection

**Parameters** **collectionID** (*str*) – a Zotero library collection ID

**Return type** list of dicts

`Zotero.`**`all_collections`**`([`*collectionID*`])`

Returns either all collections and sub-collections in a flat list, or, if a collection ID is specified, that collection and all of its sub-collections. This method can be called at any collection "depth".

**Parameters** **`collectionID`** (*str*) – a Zotero library collection ID (optional)

**Return type** list of dicts

Example of returned collection data:

```
[{u'data': {u'key': u'5TSDXJG6',
            u'name': u'Critical GIS',
            u'parentCollection': False,
            u'relations': {},
            u'version': 778},
  u'key': u'5TSDXJG6',
  u'library': {u'id': 436,
               u'links': {u'alternate': {u'href': u'https://www.zotero.org/
↪urschrei',
                                         u'type': u'text/html'}},
               u'name': u'urschrei',
               u'type': u'user'},
  u'links': {u'alternate': {u'href': u'https://www.zotero.org/urschrei/
↪collections/5TSDXJG6',
                            u'type': u'text/html'},
             u'self': {u'href': u'https://api.zotero.org/users/436/
↪collections/5TSDXJG6',
                       u'type': u'application/json'}},
  u'meta': {u'numCollections': 0, u'numItems': 1},
  u'version': 778}]
```

## Retrieving groups

`Zotero.`**`groups`**`([`*search/request parameters*`])`

Retrieve the Zotero group data to which the current library_id and api_key has access

**Return type** list of dicts

Example of returned group data:

```
[{u'data': {u'description': u'',
            u'fileEditing': u'admins',
            u'hasImage': 1,
            u'id': 169947,
            u'libraryEditing': u'admins',
            u'libraryReading': u'members',
            u'members': [1177919, 1408658],
            u'name': u'smart_cities',
            u'owner': 436,
            u'type': u'Private',
            u'url': u'',
            u'version': 0},
  u'id': 169947,
  u'links': {u'alternate': {u'href': u'https://www.zotero.org/groups/169947',
                            u'type': u'text/html'},
             u'self': {u'href': u'https://api.zotero.org/groups/169947',
```

```
                    u'type': u'application/json'}},
 u'meta': {u'created': u'2013-05-22T11:22:46Z',
           u'lastModified': u'2013-05-22T11:26:50Z',
           u'numItems': 817},
 u'version': 0}]
```

## Retrieving Tags

Zotero.**tags**([*search/request parameters*])
Returns a library's tags

**Return type** list of strings

Zotero.**item_tags**(*itemID*[, *search/request parameters*])
Returns tags from a specific item

**Parameters** **itemID** (*str*) – a valid Zotero library Item ID

**Return type** list of strings

Example of returned tag data:

```
['Authority in literature', 'Errata']
```

## Retrieving Version Information

The Zotero API recommends requesting version information for all (or all changed) items and collections when implementing syncing. The following convenience methods (which by default return an unlimited number of responses) simplify this process.

The return values of these methods associate each item / collection with the last version (or greater) at which the item / collection was modified. By passing the keyword argument `since=versionNum` only items / collections which have been modified since `versionNum` are included in the response. Thus, an application which previously sucessfully synced with the server at `versionNum` can use these methods to determine which items / collections need to be retrieved from the server.

Zotero.**item_versions**([*search/request parameters*])
Returns a dict containing version information for items in the library

**Return type** dict: string -> integer

Zotero.**collection_versions**(*itemID*[, *search/request parameters*])
Returns a dict containing version information for collections in the library

**Return type** dict: string -> integer

Example of returned version data:

```
{'C9KW275P': 3915, 'IB489TKM': 4025 }
```

## Full–Text Content

These methods allow the retrieval of full–text content for given library items

`Zotero.`**`new_fulltext`**(*since*)

Returns a dict containing item keys and library versions newer than `since` (a library version string, e.g. `"1085"`)

**rtype** dict: string -> integer

Example of returned data:

```
{
    u'229QED6I': 747,
    u'22TGJFS2': 769,
    u'23SZWREM': 764
}
```

`Zotero.`**`fulltext_item`**(*itemID*[, *search/request parameters* ])

Returns a dict containing full-text data for the given attachment item. `indexedChars` and `totalChars` are used for text documents, while `indexedPages` and `totalPages` are used for PDFs.

Example of returned data:

```
{
"content": "This is full-text content.",
"indexedPages": 50,
"totalPages": 50
}
```

`Zotero.`**`set_fulltext`**(*itemID*, *payload*)

Set full-text data for an item

**rtype** boolean

`itemID` should correspond to an existing attachment item.

`payload`: a dict containing three keys:

`content`: the full-text content, and either

For text documents, `indexedChars` and `totalChars` OR

For PDFs, `indexedPages` and `totalPages`.

Example payload:

```
{
"content": "This is full-text content.",
"indexedPages": 50,
"totalPages": 50
}
```

## The `follow()`, and `everything()` methods

These methods (currently experimental) aim to make Pyzotero a little more RESTful. Following any Read API call which can retrieve **multiple items**, calling `follow()` will repeat that call, but for the next *x* number of items, where *x* is either a number set by the user for the original call, or 50 by default. Each subsequent call to `follow()` will extend the offset.

`Zotero.`**`follow`**()

Example:

```
from pyzotero import zotero
zot = zotero.Zotero(library_id, library_type, api_key)
# only retrieve a single item
# this will retrieve the most recently added/modified top-level item
first_item = zot.top(limit=1)
# now we can start retrieving subsequent items
next_item = zot.follow()
third_item = zot.follow()
```

Zotero.**everything**()

Example:

```
from pyzotero import zotero
zot = zotero.Zotero(library_id, library_type, api_key)
# retrieve all top-level items
toplevel = zot.everything(zot.top())
```

The `everything()` method should work with all Pyzotero Read API calls which can return multiple items, but has not yet been extensively tested. Feedback is welcomed.

## Related generator methods

The *Zotero.iterfollow()* and *Zotero.makeiter()* methods are available for users who prefer to work directly with generators:

Zotero.**iterfollow**()

**Return type** a generator over the *follow()* method.

Example:

```
z = zot.top(limit=5)
lazy = zot.iterfollow()
lazy.next() # the next() call has returned the next five items
```

Zotero.**makeiter**(*API call*)

Returns a generator over a Read API method

**Parameters** **API call** (*function*) – a Pyzotero Read API method capable of returning multiple items

**Return type** generator

Example:

```
gen = zot.makeiter(zot.top(limit=5))
gen.next() # this will return the first five items
gen.next() # this will return the next five items
```

**Warning:** The `follow()`, `everything()` and `makeiter()` methods are only valid for methods which can return multiple library items. For instance, you cannot use `follow()` after an `item()` call. The generator methods will raise a `StopIteration` error when all available items retrievable by your chosen API call have been exhausted.

## Retrieving item counts

If you wish to retrieve item counts for subsets of a library, you can use the following methods:

`Zotero.num_items()`
Returns the count of top-level items in the library

**Return type** int

`Zotero.num_collectionitems(collectionID)`
Returns the count of items in the specified collection

**Return type** int

`Zotero.num_tagitems(tag)`
Returns the count of items for the specified tag

**Return type** int

## Retrieving last modified version

If you wish to retrieve the last modified version of a library, you can use the following method:

`Zotero.last_modified_version()`
Returns the last modified version of the library

**Return type** int

## Search / Request Parameters for Read API calls

Additional parameters may be set on Read API methods **following any required parameters**, or set using the *Zotero.add_parameters()* method detailed below.

The following two examples produce the same result:

```
# set parameters on the call itself
z = zot.top(limit=7, start=3)

# set parameters using explicit method
zot.add_parameters(limit=7, start=3)
z = zot.top()
```

The following parameters are are **optional**.

**You may also set a search term here, using the 'itemType', 'q', 'qmode', or 'tag' parameters**.

This area of the Zotero Read API is under development, and may change frequently. See the API documentation for the most up-to-date details of search syntax usage and export format details.

`Zotero.add_parameters([format=None, itemKey=None, itemType=None, q=None, qmode=None, since=None, tag=None, sort=None, direction=None, limit=None, start=None[, content=None[, style=None]]])`

**Parameters**

- **format** (*str*) – "atom", "bib", "json", "keys", "versions". Pyzotero retrieves and decodes JSON responses by default

- **itemKey** (*str*) – A comma-separated list of item keys. Valid only for item requests. Up to 50 items can be specified in a single request

Search parameters:

**Parameters**

- **itemType** (*str*) – item type search. See the Search Syntax for details
- **q** (*str*) – Quick search. Searches titles and individual creator fields by default. Use the `qmode` parameter to change the mode. Currently supports phrase searching only
- **qmode** (*str*) – Quick search mode. To include full-text content in the search, use `everything`. Defaults to `titleCreatorYear`. Searching of other fields will be possible in the future
- **since** (*int*) – default `0`. Return only objects modified after the specified library version
- **tag** (*str*) – tag search. See the Search Syntax for details. More than one tag may be passed by passing a list of strings. These are treated as `AND` search terms.

The following parameters can be used for search requests:

**Parameters**

- **sort** (*str*) – The name of the field by which entries are sorted: (`dateAdded`, `dateModified`, `title`, `creator`, `type`, `date`, `publisher`, `publicationTitle`, `journalAbbreviation`, `language`, `accessDate`, `libraryCatalog`, `callNumber`, `rights`, `addedBy`, `numItems`, (tags))
- **direction** (*str*) – `asc` or `desc`
- **limit** (*int*) – 1 – 100 or None
- **start** (*int*) – 1 – total number of items in your library or None

If you wish to retrieve citation or bibliography entries, use the following parameters:

**Parameters**

- **content** (*str*) – 'bib', 'html', or one of the export formats (see below). If 'bib' is passed, you may **also** pass:
- **style** (*str*) – Any valid CSL style in the Zotero style repository

**Return type** list of HTML strings or None.

---

**Note:** Any parameters you set will be valid **for the next call only**. Any parameters set using `add_parameters()` will be overridden by parameters you pass in the call itself.

---

A note on the `content` and `style` parameters:

Example:

```
zot.add_parameters(content='bib', style='mla')
```

If these are set, the return value is a list of UTF-8 formatted HTML `div` elements, each containing an item:

`['<div class="csl-entry">(content)</div>'].`

You may also set `content='citation'` if you wish to retrieve citations. Similar to `bib`, the result will be a list of one or more HTML `span` elements.

If you select one of the available export formats as the `content` parameter, pyzotero will in most cases return a list of unicode strings in the format you specified. The exception is the `csljson` format, which is parsed into a list of dicts. Please note that you must provide a `limit` parameter if you specify one of these export formats. Multiple simultaneous retrieval of particular formats, e.g. `content="json,coins"` is not currently supported.

If you set `format='keys'`, a newline-delimited string containing item keys will be returned

CHAPTER 4

# Write API Methods

## Item Methods

Zotero.**item_types**()
Returns a dict containing all available item types

**Return type** dict

Zotero.**item_fields**()
Returns a dict of all available item fields

**Return type** dict

Zotero.**item_creator_types**(*itemtype*)
Returns a dict of all valid creator types for the specified item type

**Parameters** **itemtype** (*str*) – a valid Zotero item type. A list of available item types can be obtained by the use of *item_types()*

**Return type** dict

Zotero.**creator_fields**()
Returns a dict containing all localised creator fields

**Return type** dict

Zotero.**item_type_fields**(*itemtype*)
Returns all valid fields for the specified item type

**Parameters** **itemtype** (*str*) – a valid Zotero item type. A list of available item types can be obtained by the use of *item_types()*

**Return type** list of dicts

Zotero.**item_template**(*itemtype*)
Returns an item creation template for the specified item type

**Parameters** **itemtype** (*str*) – a valid Zotero item type. A list of available item types can be obtained by the use of *item_types()*

**Return type** dict

## Creating items

Zotero.**create_items**(*items*[, *parentid*, *last_modified*])
Create Zotero library items

**Parameters**

- **items** (*list*) – one or more dicts containing item data
- **parentid** (*str*) – A Parent item ID. This will cause the item(s) to become the child items of the given parent ID
- **last_modified** (*str/int*) – If not None will set the value of the If-Unmodified-Since-Version header.

**Return type** list of dicts

Returns a copy of the created item(s), if successful. Use of *item_template()* is recommended in order to first obtain a dict with a structure which is known to be valid.

Before calling this method, the use of *check_items()* is encouraged, in order to confirm that the item to be created contains only valid fields.

Note that if any items contain a key field matching an existing item on the server it will be updated (any properties not in the dict will be left unmodified).

Example:

```
template = zot.item_template('book')
template['creators'][0]['firstName'] = 'Monty'
template['creators'][0]['lastName'] = 'Cantsin'
template['title'] = 'Maris Kundzins: A Life'
resp = zot.create_items([template])
```

If successful, `resp` will be a dict containing the creation status of each item:

```
{'failed': {}, 'success': {'0': 'ABC123'}, 'unchanged': {}}
```

Zotero.**update_item**(*item*[, *last_modified*])
Update an item in your library

**Parameters**

- **item** (*dict*) – a dict containing item data. Fields not in item will be left unmodified.
- **last_modified** (*str/int*) – If not `None`, will set the value of the If-Unmodified-Since-Version header. If unspecified/None then If-Unmodified-Since-Version will be set to the version property of item.

**Return type** Boolean

Will return `True` if the request was successful, or will raise an error.

Example:

```
i = zot.items()
# see above for example of returned item structure
# modify the latest item which was added to your library
i[0]['data']['title'] = 'The Sheltering Sky'
```

```
i[0]['data']['creators'][0]['firstName'] = 'Paul'
i[0]['data']['creators'][0]['lastName'] = 'Bowles'
zot.update_item(i[0])
```

Zotero.**check_items**(*items*)

Check whether items to be created on the server contain only valid keys. This method first creates a set of valid keys by calling *item_fields()*, then compares the user-created dicts to it. If any keys in the user-created dicts are unknown, a KeyError exception is raised, detailing the invalid fields.

**Parameters** **items** (*list*) – one or more dicts containing item data

**Return type** List. Each list item is a valid dict containing item data.

## Uploading files

**Warning:** Attachment methods are in beta.

Zotero.**attachment_simple**(*files*[, *parentid*])

Create one or more file attachment items.

**Parameters**

- **files** (*list*) – a list containing one or more file paths: ['/path/to/file/file.pdf', ... ]
- **parentid** (*string*) – a library Item ID. If this is specified, attachments will be created as child items of this ID.

**Return type** Dict. Showing status of each requested upload.

Zotero.**attachment_both**(*files*[, *parentid*])

Create one or more file attachment items, specifying names for uploaded files

**Parameters**

- **files** (*list*) – a list containing one or more lists or tuples in the following format: (file name, file path)
- **parentid** (*string*) – a library Item ID. If this is specified, attachments will be created as child items of this ID.

**Return type** Dict. Showing status of each requested upload.

Zotero.**upload_attachments**(*attachments*[, *parentid*, *basedir=None*])

Upload files to their corresponding attachments. If the attachments lack the key property they are assumed not to exist and will be created. The parentid parameter is **not compatible** with existing attachments. In order for uploads to succeed, the filename parameter of each attachment must resolve.

**Parameters**

- **attachments** (*list*) – A list of dicts representing zotero imported files which may or may not already have their key fields filled in.
- **parentid** (*string*) – a library Item ID. If this is specified and key fields are not included, attachments will be created as child items of this ID.

- **basedir** (*string/path*) – A string or path object to which the filenames specified in attachments will be evaluated relative to. If unspecified the filenames are evaluated as they are.

**Return type** Dict. Showing status of each requested upload.

```
# example of the return type
{
    'success': [attach1, attach2...],
    'failure': [attach3, attach4...],
    'unchanged': [attach4, attach5...]
}
```

---

**Note:** unlike the space-saving responses from the server, the return value here eschews the complex index / key lookup and simply passes back the `imported_file` item template populated with keys (if created successfully or passed in) corresponding to each result. This is the return type for all of these methods.

---

### Deleting items

`Zotero.`**delete_item**(*item*[, *last_modified*])

Delete one or more items from your library

**Parameters**

- **item** (*list*) – a list of one or more dicts containing item data. You must first retrieve the item(s) you wish to delete, as `version` data is required.
- **last_modified** (*str/int*) – If not `None`, will set the value of the If-Unmodified-Since-Version header.

### Deleting tags

`Zotero.`**delete_tags**(*tag_a*[, *tag* ...])

Delete one or more tags from your library

**Parameters** **tag** (*string*) – the tag(s) you'd like to delete

You may also pass a list using `zot.delete_tags(*[tag_list])`

## Adding tags

`Zotero.`**add_tags**(*item*, *tag*[, *tag* ...])

Add one or more tags to an item, and update it on the server

**Parameters**

- **item** (*dict*) – a dict containing item data
- **tag** (*string*) – the tag(s) you'd like to add to the item

**Return type** list of dicts

You may also pass a list using `zot.add_tags(item, *[tag_list])`

Example:

```
z = zot.top(limit=1)
# we've now retrieved the most recent top-level item
updated = zot.add_tags(z[0], 'tag1', 'tag2', 'tag3')
# updated now contains a representation of the updated server item
```

## Collection Methods

Zotero.**create_collections**(*dicts*[, *last_modified*])
Create a new collection in the Zotero library

**Parameters**

- **dicts** (*list*) – list of dicts each containing the key `name`, with each value being a new collection name you wish to create. Each dict may optionally contain a `parent` key, the value of which is the ID of an existing collection. If this is set, the collection will be created as a child of that collection.
- **last_modified** (*str/int*) – If not None will set the value of the If-Unmodified-Since-Version header.

**Return type** list of dicts

**Return type** Boolean

Zotero.**create_collections**(*dicts*[, *last_modified*])
Alias for create_collections to preserve backward compatibility

Zotero.**addto_collection**(*collection*, *item*)
Add the specified item(s) to the specified collection

**Parameters**

- **collection** (*str*) – a collection key
- **item** (*dict*) – an item dict retrieved using an API call

**Return type** Boolean

Collection keys can be obtained by a call to *collections()* (see details above).

Zotero.**deletefrom_collection**(*collection*, *item*)
Remove the specified item from the specified collection

**Parameters**

- **collection** (*str*) – a collection key
- **item** (*dict*) – a dict containing item data

**Return type** Boolean

See the *delete_item()* example for multiple-item removal.

Zotero.**update_collection**(*collection , last_modified]*)
Update an existing collection name

**Parameters** **collection** (*dict*) – a dict containing collection data, previously retrieved using one of the Collections calls (e.g. *collections()*)

**Return type** Boolean

Example:

```
# get existing collections, which will return a list of dicts
c = zot.collections()
# rename the last collection created in the library
c[0]['name'] = 'Whither Digital Humanities?'
# update collection name on the server
zot.update_collection(c[0])
```

Zotero.**delete_collection**(*collection*[, *last_modified*])

Delete a collection from the Zotero library

**Parameters**

- **collection** (*dict*) – a dict containing collection data, previously retrieved using one of the Collections calls (e.g. *collections()*). Alternatively, you may pass a **list** of collection dicts.
- **last_modified** (*str/int*) – If not None will set the value of the If-Unmodified-Since-Version header.

**Return type** Boolean

CHAPTER 5

# Notes

Most Read API methods return **lists** of **dicts** or, in the case of tag methods, **lists** of **strings**. Most Write API methods return either `True` if successful, or raise an error. See `zotero_errors.py` for a full listing of these.

**Warning:** URL parameters will supersede API calls which should return e.g. a single item: `https://api.zotero.org/users/436/items/ABC?start=50&limit=10` will return 10 items beginning at position 50, even though `ABC` does not exist. Be aware of this, and don't pass URL parameters which do not apply to a given API method.

CHAPTER 6

# License

Pyzotero is licensed under the MIT license.

CHAPTER 7

# Cat Picture

This is The Grinch.

Fig. 7.1: *Orangecat*

# Python Module Index

## Z

zotero, 1

# Index


## A

add_parameters() (zotero.Zotero method), 15
add_tags() (zotero.Zotero method), 22
addto_collection() (zotero.Zotero method), 23
all_collections() (zotero.Zotero method), 11
attachment_both() (zotero.Zotero method), 21
attachment_simple() (zotero.Zotero method), 21

## C

check_items() (zotero.Zotero method), 21
children() (zotero.Zotero method), 8
collection() (zotero.Zotero method), 10
collection_items() (zotero.Zotero method), 8
collection_items_top() (zotero.Zotero method), 8
collection_versions() (zotero.Zotero method), 12
collections() (zotero.Zotero method), 10
collections_sub() (zotero.Zotero method), 10
collections_top() (zotero.Zotero method), 10
create_collections() (zotero.Zotero method), 23
create_items() (zotero.Zotero method), 20
creator_fields() (zotero.Zotero method), 19

## D

delete_collection() (zotero.Zotero method), 24
delete_item() (zotero.Zotero method), 22
delete_tags() (zotero.Zotero method), 22
deleted() (zotero.Zotero method), 8
deletefrom_collection() (zotero.Zotero method), 23
dump() (zotero.Zotero method), 10

## E

everything() (zotero.Zotero method), 14

## F

file() (zotero.Zotero method), 9
follow() (zotero.Zotero method), 13
fulltext_item() (zotero.Zotero method), 13

## G

get_subset() (zotero.Zotero method), 8
groups() (zotero.Zotero method), 11

## I

item() (zotero.Zotero method), 8
item_creator_types() (zotero.Zotero method), 19
item_fields() (zotero.Zotero method), 19
item_tags() (zotero.Zotero method), 12
item_template() (zotero.Zotero method), 19
item_type_fields() (zotero.Zotero method), 19
item_types() (zotero.Zotero method), 19
item_versions() (zotero.Zotero method), 12
items() (zotero.Zotero method), 7
iterfollow() (zotero.Zotero method), 14

## K

key_info() (zotero.Zotero method), 7

## L

last_modified_version() (zotero.Zotero method), 15

## M

makeiter() (zotero.Zotero method), 14

## N

new_fulltext() (zotero.Zotero method), 13
num_collectionitems() (zotero.Zotero method), 15
num_items() (zotero.Zotero method), 15
num_tagitems() (zotero.Zotero method), 15

## S

set_fulltext() (zotero.Zotero method), 13

## T

tags() (zotero.Zotero method), 12
top() (zotero.Zotero method), 7
trash() (zotero.Zotero method), 8

## U

update_collection() (zotero.Zotero method), 23
update_item() (zotero.Zotero method), 20
upload_attachments() (zotero.Zotero method), 21

## Z

Zotero (class in zotero), 6
zotero (module), 1