python-documentcloud Documentation

Release 2.0.0

Los Angeles Times Data Desk

Mar 11, 2021

Contents

A simple Python wrapper for the DocumentCloud API

CHAPTER 1

Features

- Retrieve and edit documents and projects, both public and private, from documentcloud.org
- Upload PDFs into your documentcloud.org account and organize them into projects
- Download text and images extracted from your PDFs by DocumentCloud

CHAPTER 2

Documentation

2.1 Getting started

This tutorial will walk you through the process of installing python-documentcloud and making your first requests.

2.1.1 Installation

Provided that you have pip installed, you can install the library like so:

```
$ pip install documentcloud
```

2.1.2 Creating a client

Before you can interact with DocumentCloud, you first must import the library and initialize a client to talk with the site on your behalf.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud()
```

Since we didn't provide any log-in credentials, the client above will only be able to access published documents. If have an account at DocumentCloud and want to use that, you can provide the crendentials to the client.

```
>>> client = DocumentCloud(USERNAME, PASSWORD)
```

You can also specify a custom uri if you have installed your own version of DocumentCloud

```
>>> client = DocumentCloud(USERNAME, PASSWORD, base_uri="https://your.documentcloud.
↪domain/api/", auth_uri="https://your.account.server.domain/api/")
```

2.1.3 Searching for documents

You can now you use client to interact with DocumentCloud. A search for documents about journalist Ruben Salazar would look like this:

```
>>> obj_list = client.documents.search("Ruben Salazar")
>>> # Let's grab the first one and look at it
>>> obj = obj_list[0]
>>> obj
<Document: Final OIR Report>
```

2.1.4 Interacting with a document

Once you have you hands on a document object, you can interact with the metadata stored at documentcloud.org. Here's a sample:

```
>>> print obj.title
Final OIR Report
>>> print obj.id
71072
>>> print obj.contributor_organization
Los Angeles Times
>>> print obj.canonical_url
http://www.documentcloud.org/documents/71072-oir-final-report.html
```

You can even download the PDF, page images and full text.

```
>>> obj.large_image_url
...
>>> obj.large_image
...
>>> obj.full_text
...
>>> obj.pdf
...
```

2.1.5 Uploading a document

You can upload a PDF document from your local machine to documentcloud.org. Here's how:

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj = client.documents.upload("/home/ben/pdfs/myfile.pdf")
```

And you don't have to provide a path, you can also upload a file object.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> pdf = open("/home/ben/pdfs/myfile.pdf", "rb")
>>> obj = client.documents.upload(pdf)
```

You can also provide URLs that link to PDFs, if that's the kind of thing you're into.

```
>>> client.documents.upload("http://ord.legistar.com/Chicago/attachments/e3a0cbcb-
↪044d-4ec3-9848-23c5692b1943.pdf")
```

2.1.6 Interacting with a newly uploaded public document

When you first upload a document, your local document object will not reflect some of the metadata and processing that happens in the first few seconds it is on the server. Documents set to public will be shown as private during that short interval. To interact with a document as soon as it is available, you can write a short loop to check whether it is ready.

First upload the document as normal.

```
>>> import time
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj = client.documents.upload("/home/ben/pdfs/myfile.pdf", access='public')
```

Then refresh your local document object from the server. If it is does not show up as public, then it is still processing, and you'll have to check again.

```
>>> obj = client.documents.get(obj.id)
>>> while obj.status != 'success':
>>>     time.sleep(5)
>>>     obj = client.documents.get(obj.id)
```

2.1.7 Uploading a directory of documents as a project

Here's how to upload a directory full of documents and add them all to a new project. Be warned, this will upload any documents in directories inside the path you specify.

```
>>> # Connect to documentcloud
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> # Create the project
>>> project, created = client.projects.get_or_create_by_title("Groucho Marx's FBI file
↪")
>>> # Upload all the pdfs
>>> obj_list = client.documents.upload_directory('/home/ben/pdfs/groucho_marx/')
>>> # Add the newly created documents to the project
>>> project.document_list = obj_list
>>> # Save the changes to the project
>>> project.put()
```

2.1.8 Uploading a PDF from a URL

How to read a PDF document from a URL on the World Wide Web and upload it to DocumentCloud without saving it to your local hard drive.

```
>>> from documentcloud import DocumentCloud
>>> url = "http://myhost.org/interesting-doc.pdf"
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> # Upload the specified URL to the given client
>>> obj = client.documents.upload(url)
```

2.2 Documents

Methods for drawing down, editing and uploading data about documents.

2.2.1 DocumentClient

`class documentcloud.documents.DocumentClient`

The document client gives access to retrieval and uploading of documents. It is generally accessed as `client.documents`.

`all(self, **params)`

An alias for `list()`.

`get(id_, expand=None)`

Return the document with the provided DocumentCloud identifer.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> client.documents.get(71072)
<Document: Final OIR Report>
```

The identifier may be just the numeric ID (71072, preferred), the old style ID-slug (71072-oir-final-report), or the new style slug-id (oir-final-report-71072).

Setting expand allows for the user or organization details to be fetched with the document, instead of requiring a separate API request to fetch them. Set expand to a list of the values you would like expanded.

```
>>> client.documents.get(71072, expand=["user"])
>>> client.documents.get(71072, expand=["user", "organization"])
```

`list(self, **params)`

Return a list of all documents, possibly filtered by the given parameters. Please see the full API documentation for available parameters.

`search(query, **params)`

Return a list of documents that match the provided query.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud()
>>> obj_list = client.documents.search('Ruben Salazar')
>>> obj_list[0]
<Document: Final OIR Report>
```

The params may be set to any parameters that the search end point takes. Please see the full search documentation for query syntax and available parameters.

`upload(pdf, **kwargs)`

Upload a PDF to DocumentCloud. You must be authorized to do this. Returns the object representing the new record you've created. You can submit either a file path or a file object.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> new_id = client.documents.upload("/home/ben/test.pdf", "Test PDF")
>>> # Now fetch it
>>> client.documents.get(new_id)
<Document: Test PDF>
```

You can also use URLs which link to PDFs, if that's the kind of thing you want to do.

```
>>> upload("http://ord.legistar.com/Chicago/attachments/e3a0cbcb-044d-
↪4ec3-9848-23c5692b1943.pdf")
```

You may set the `kwargs` to any of the writable attributes as described in *`documentcloud.documents.Document.put()`*. Additionally, you may set `force_ocr` in order to force OCR to take place even if the document has embedded text, as well as either `project` to the ID of a project to upload the document into, or `projects`, a list of project IDs to upload the document into. If you are uploading a non-PDF document type, you must set `original_extension` to the extension of the file type, such as `docx` or `jpg`.

`upload_directory`(*path*, *handle_errors=False*, ***kwargs*)

Searches through the provided path and attempts to upload all the PDFs it can find. Metadata, which accepts the same keywords as *`upload()`*, provided to the other keyword arguments will be recorded for all uploads (except for title which will be set based on the filename). Returns a list of document objects that are created. Be warned, this will upload any documents in directories inside the path you specify. The handle_errors parameter will catch and print errors generated by the network request or the DocumentCloud API, log them, and try to continue processing. This might be useful if you are uploading a very large directory and do not want temporary network problems to stop the entire upload.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj_list = client.documents.upload_directory('/home/ben/pdfs/groucho_marx/
↪')
```

2.2.2 Document

class `documentcloud.documents.`**`Document`**

An individual document, as obtained by the *`documentcloud.documents.DocumentClient`*.

`put`()

Save changes to a document back to DocumentCloud. You must be authorized to make these changes. Only the *`access`*, *`data`*, *`description`*, *`language`*, *`related_article`*, *`published_url`*, *`source`*, and *`title`*, attributes may be edited.

```
>>> # Grab a document
>>> obj = client.documents.get('71072')
>>> print obj.title
Draft OIR Report
>>> # Change its title
>>> obj.title = "Brand new title"
>>> print obj.title
Brand New Title
>>> # Save those changes
>>> obj.put()
```

`delete`()

Delete a document from DocumentCloud. You must be authorized to make these changes.

```
>>> obj = client.documents.get('71072-oir-final-report')
>>> obj.delete()
```

`process`()

This will re-process the document. Useful if there was an intermittent error.

save()

An alias for *put()* that saves changes back to DocumentCloud.

access

The privacy level of the resource within the DocumentCloud system. It will be either `public`, `private` or `organization`, the last of which means the is only visible to members of the contributors organization. Can be edited and saved with a put command.

annotations

A client to access and update the annotations on the document. See `Annotation` for more information.

asset_url

The base URL to obtain the static assets for this document. See the API documentation for more details.

canonical_url

The URL where the document is hosted at documentcloud.org.

contributor

The user who originally uploaded the document.

contributor_organization

The organizational affiliation of the user who originally uploaded the document.

created_at

The date and time that the document was created, in Python's datetime format.

data

A dictionary containing supplementary data linked to the document. This can be any old thing. It's useful if you'd like to store additional metadata. Can be edited and saved with a put command.

```
>>> obj = client.documents.get('83251-fbi-file-on-christopher-biggie-smalls-
↪wallace')
>>> obj.data
{'category': 'hip-hop', 'byline': 'Ben Welsh', 'pub_date': datetime.date(2011,
↪ 3, 1)}
```

Keys must be strings and only contain alphanumeric characters.

description

A summary of the document. Can be edited and saved with a put command.

edit_access

A boolean indicating whether or not you have the ability to save this document.

file_hash

A hash representation of the raw PDF data as a hexadecimal string.

```
>>> obj = client.documents.get('1021571-lafd-2013-hiring-statistics')
>>> obj.file_hash
'872b9b858f5f3e6bb6086fec7f05dd464b60eb26'
```

You could recreate this hexadecimal hash yourself using the SHA-1 algorithm.

```
>>> import hashlib
>>> hashlib.sha1(obj.pdf).hexdigest()
'872b9b858f5f3e6bb6086fec7f05dd464b60eb26'
```

full_text

Returns the full text of the document, as extracted from the original PDF by DocumentCloud. Results may vary, but this will give you what they got.

```
>>> obj = client.documents.get('71072-oir-final-report')
>>> obj.full_text
"Review of the Los Angeles County Sheriff's\nDepartment's Investigation
into the\nHomicide of Ruben Salazar\nA Special Report by the\nLos
Angeles County Office of Independent Review\n ...
```

`full_text_url`

Returns the URL that contains the full text of the document, as extracted from the original PDF by DocumentCloud.

`get_page_text` (*page*)

Submit a page number and receive the raw text extracted from it by DocumentCloud.

```
>>> obj = client.documents.get('1088501-adventuretime-alta')
>>> txt = obj.get_page_text(1)
# Let's print just the first line
>>> print txt.decode().split("\n")[0]
STATE OF CALIFORNIA- HEALTH AND HUMAN SERVICES AGENCY
```

`id`

The unique identifer of the document in DocumentCloud's system. This is a number. `83251`

`language`

The three character code for the language this document is in.

`large_image`

Returns the binary data for the "large" sized image of the document's first page. If you would like the data for some other page, pass the page number into `get_large_image(page)`.

`large_image_url`

Returns a URL containing the "large" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `get_large_image_url(page)`.

`large_image_url_list`

Returns a list of URLs for the "large" sized image of every page in the document.

`mentions`

When the document has been retrieved via a search, this returns a list of places the search keywords appear in the text. The data are modeled by their own Python class, *`documentcloud.documents.Mention`*.

```
>>> obj_list = client.documents.search('Christopher Wallace')
>>> obj = obj_list[0]
>>> obj.mentions
[<Mention: Page 2>, <Mention: Page 3> ....
```

`normal_image`

Returns the binary data for the "normal" sized image of the document's first page. If you would like the data for some other page, pass the page number into `get_normal_image(page)`.

`normal_image_url`

Returns a URL containing the "normal" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `get_normal_image_url(page)`.

`normal_image_url_list`

Returns a list of URLs for the "normal" sized image of every page in the document.

`organization`

The *`documentcloud.organizations.Organization`* which owns this document. This will require an additional API call unless you specify *"organization"* in the *expand* parameter when fetching this document.

organization_id
The ID for the organization which owns this document

page_count
Alias for `pages`.

page_spec
The page spec is a compressed string that lists dimensions in pixels for every page in a document. Refer to ListCrunch for the compression format. For example, *612.0x792.0:0-447*

pages
The number of pages in the document.

pdf
Returns the binary data for document's original PDF file.

pdf_url
Returns a URL containing the binary data for document's original PDF file.

projects
Returns a list of IDs for the projects this document is in.

published_url
Returns an URL outside of documentcloud.org where this document has been published.

related_article
Returns an URL for a news story related to this document.

sections
A client to access and update the sections on the document. See `documentcloud.sections.Section` for more information.

slug
Returns the document's slug. A slug is a URL friendly version of the title.

small_image
Returns the binary data for the "small" sized image of the document's first page. If you would like the data for some other page, pass the page number into `get_small_image(page)`.

small_image_url
Returns a URL containing the "small" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `get_small_image_url(page)`.

small_image_url_list
Returns a list of URLs for the "small" sized image of every page in the document.

source
The original source of the document. Can be edited and saved with a put command.

status
This is the status of the document. Possible statuses include:

- *success*: The document has been succesfully processed
- *readable*: The document is currently processing, but is readable during the operation
- *pending*: The document is processing and not currently readable
- *error*: There was an [error](#errors) during processing
- *nofile*: The document was created, but no file was uploaded yet

thumbnail_image
Returns the binary data for the "thumbnail" sized image of the document's first page. If you would like the data for some other page, pass the page number into `get_thumbnail_image(page)`.

thumbnail_image_url
Returns a URL containing the "thumbnail" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `get_small_thumbnail_url(page)`.

thumbnail_image_url_list
Returns a list of URLs for the "small" sized image of every page in the document.

title
The name of the document. Can be edited and saved with a put command.

updated_at
The date and time that the document was last updated, in Python's datetime format.

user
The *documentcloud.users.User* which owns this document. This will require an additional API call unless you specify *"user"* in the *expand* parameter when fetching this document.

user_id
The ID for the user which owns this document

2.2.3 Mentions

class documentcloud.documents.**Mention**
Mentions of a search keyword found in one of the documents.

page
The page where the mention occurs.

text
The text surrounding the mention of the keyword.

2.3 Projects

Methods for drawing down, editing and uploading data about DocumentCloud projects. A project is a group of documents.

2.3.1 ProjectClient

class documentcloud.projects.**ProjectClient**
The project client gives access to retrieval and creation of projects. It is generally accessed as `client.projects`.

all(***params*)
Return all projects for the authorized DocumentCloud account. Please see the full API documentation for available parameters.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> obj_list = client.projects.all()
>>> obj_list[0]
<Project: Ruben Salazar>
```

create (*title*, *description=""*, *private=True*, *document_ids=None*)

Create a new project on DocumentCloud. You must be authorized to do this. Returns the object representing the new record you've created.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> obj = client.projects.create("New project")
>>> obj
<Project: New project>
```

get (*id=None*, *title=None*)

Return the project with the provided DocumentCloud identifer. You can retrieve projects using either the *id* or *title*.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> # Fetch using the id
>>> obj = client.projects.get(id='816')
>>> obj
<Project: The Ruben Salazar Files>
>>> # Fetch using the title
>>> obj = client.projects.get(title='The Ruben Salazar Files')
>>> obj
<Project: The Ruben Salazar Files>
```

get_or_create_by_title (*title*)

Fetch the project with provided name, or create it if it does not exist. You must be authorized to do this. Returns a tuple. An object representing the record comes first. A boolean that reports whether or not the objects was created fresh comes second. It is true when the record was created, false when it was found on the site already.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> # The first time it will be created and added to documentcloud.org
>>> obj, created = client.projects.get_or_create_by_title("New project")
>>> obj, created
<Project: New project>, True
>>> # The second time it will be fetched from documentcloud.org
>>> obj, created = client.projects.get_or_create_by_title("New project")
>>> obj, created
<Project: New project>, False
```

get_by_id (*id*)

Return the project with the provided id. Operates the same as *client.projects.get*.

get_by_title (*title*)

Return the project with the provided title. Operates the same as *client.projects.get*.

list (***params*)

List all projects. Please see the full API documentation for available parameters.

2.3.2 Project

class documentcloud.projects.**Project**

An individual project, as obtained by the `documentcloud.projects.ProjectClient`.

put()

Save changes to a project back to DocumentCloud. You must be authorized to make these changes. Only the *description*, *document_list*, *private*, and *title*, attributes may be edited.

```
>>> obj = client.projects.get('816')
>>> obj.title = "Brand new title"
>>> obj.put()
```

delete()

Delete a project from DocumentCloud. You must be authorized to make these changes.

```
>>> obj = client.projects.get('816')
>>> obj.delete()
```

save()

An alias for *put()* that saves changes back to DocumentCloud.

created_at

The date and time when this project was created

description

A summary of the project. Can be edited and saved with a put command.

document_ids

A list that contains the unique identifier of the documents assigned to this project. Cannot be edited. Edit the document_list instead.

```
>>> obj = client.projects.get('816')
>>> obj.document_ids
[19419, 19420, 19280, 19281, ...
```

document_list

A list that documents assigned to this project. Can be expanded by appending new documents to the list or cleared by reassigning it as an empty list and then issuing the put command.

```
>>> obj = client.projects.get('816')
>>> obj.document_list
[<Document: Times Columnist Ruben Salazar Slain by Tear-gas Missile>, <Document: Salazar's Legacy Lives On>, <Document: Cub Reporter Catches Attention of El Paso FBI>, ...
```

documents

Alias for *document_list*.

edit_access

A boolean indicating whether or not you have the ability to save this project.

get_document(*id*)

Retrieves a particular document from the project using the provided DocumentCloud identifer.

id

The unique identifer of the project in DocumentCloud's system. Typically this is a number.

private

Whether or not this project is private. Private documents in public projects will not be viewable, but setting a project to private will protect its existence from being publically viewable.

slug

The slug for the project. A slug is a URL friendly version of the title.

title
The name of the project. Can be edited and saved with a put command.

updated_at
The date and time of the last time this project was updated.

user
The ID of the `documentcloud.users.User` who created this project.

2.4 Annotations

Methods for drawing down, editing and creating annotations on documents.

2.4.1 AnnotationClient

class `documentcloud.annotations.`**AnnotationClient**
The annotation client gives access to retrieval and creation of notes on a given document. It is accessed as `doc_obj.annotations`. The annotation client supports iteration, length and indexing so that it can be used directly as a sequence of the existing notes.

```
>>> for note in doc_obj.annotations:
>>>     print(note)
>>> note_count = len(doc_obj.annotations)
>>> second_note = doc_obj.annotations[1]
```

all(*self*, ***params*)
An alias for `list()`.

create(*self*, *title*, *page_number*, *content=""*, *access="private"*, *x1=None*, *y1=None*, *x2=Non2*, *y2=Non2*)
Create a new annotation on this document. You may leave all coordinates blank to specify a full page note, or you must specify all documents in percentage of the page, a number between 0 and 1.

get(*id_*)
Return the annotation with the provided identifier.

list(*self*, ***params*)
Return a list of all annotations on this document, possibly filtered by the given parameters. Please see the full API documentation for available parameters.

2.4.2 Annotation

class `documentcloud.annotations.`**Annotation**
A note left on a document.

put()
Save changes to an annotation back to DocumentCloud. You must be authorized to make these changes. Only the `access`, `content`, `page_number`, `title`, `x1`, `x2`, `y1`, and `y2`, attributes may be edited.

delete()
Delete an annotation from DocumentCloud. You must be authorized to make these changes.

save()
An alias for `put()` that saves changes back to DocumentCloud.

access
The privacy level of the annotation within the DocumentCloud system. It will be `public`, `private`, or `organization`. `organization` will extend access to the note to any user with edit access to the document the note is attached to, including project collaborators.

content
Space for a lengthy text block that will be published below the highlighted text in the DocumentCloud design.

created_at
The date and time this annotation was created.

description
Alias for *content*.

edit_access
A boolean indicating whether or not you have the ability to save this annotation.

id
The unique identifer of the annotation in DocumentCloud's system.

location
Deprecated since version 2.0.0: The location of where the annotation appears on the document's page. Defined by *documentcloud.annotations.Location*.

organization
The ID of the organization which owns this note.

page
The page where the annotation appears.

page_number
Alias for *page*.

title
The name of the annotation, which appears in the table of contents and above the highlighted text when published by DocumentCloud.

updated_at
The date and time of when this annotation was last updated.

user
The ID of the user who created this annotation.

x1

x2

y1

y2
The coordinates for the annotation, in percentage of the page. They will be floats between 0 and 1. *x1* corresponds to the left, *x2* to the right, *y1* to the top, and *y2* to the bottom coordinate.

2.4.3 Location

class documentcloud.annotations.**Location**
The location where an *documentcloud.annotations.Annotation* is placed within a document. The coordinates are in pixels, normalized for a 700 pixel width page.

Deprecated since version 2.0.0: The new API directly exposes the top level attributes *documentcloud.annotations.Annotation.x1*, *documentcloud.annotations.Annotation.x2*, *documentcloud.annotations.Annotation.y1*, and *documentcloud.annotations.Annotation.y2*, which are in percentage of page. New code should use those instead of the location object.

bottom
The value of the bottom edge of an annotation.

left
The value of the left edge of an annotation.

right
The value of the right edge of an annotation.

top
The value of the top edge of an annotation.

2.5 Sections

Methods for drawing down, editing and creating sections of documents.

2.5.1 SectionClient

class documentcloud.sections.**SectionClient**
The section client gives access to retrieval and creation of sections on a given document. It is accessed as `doc_obj.sections`. The section client supports iteration, length and indexing so that it can be used directly as a sequence of the existing sections.

```
>>> for section in doc_obj.sections:
>>>     print(section)
>>> section_count = len(doc_obj.sections)
>>> second_section = doc_obj.sections[1]
```

all(*self*, ***params*)
An alias for *list()*.

create(*self*, *title*, *page_number*)
Create a new section on this document. There may only be one section per page.

get(*id_*)
Return the section with the provided identifier.

list(*self*, ***params*)
Return a list of all sections on this document, possibly filtered by the given parameters. Please see the full API documentation for available parameters.

2.5.2 Section

class documentcloud.sections.**Section**
Sections of the documents earmarked by users.

put()
Save changes to a section back to DocumentCloud. You must be authorized to make these changes. Only the *page_number*, and *title*, attributes may be edited.

delete()
Delete a section from DocumentCloud. You must be authorized to make these changes.

save()
: An alias for *put ()* that saves changes back to DocumentCloud.

title
: The name of the section.

page
: The page where the section begins.

page_number
: Alias to *page*.

2.6 Users

Users of the DocumentCloud service.

2.6.1 UserClient

class documentcloud.users.**UserClient**
: The user client gives access to retrieval of users. It is generally accessed as `client.users`.

all(*self*, ***params*)
: An alias for *list ()*.

get(*id_*)
: Return the user with the provided DocumentCloud identifer.

list(*self*, ***params*)
: Return a list of all users, possibly filtered by the given parameters. Please see the full API documentation for available parameters.

2.6.2 User

class documentcloud.users.**User**

avatar_url
: A URL pointing to an image representing the user

id
: The unique identifer of the user in DocumentCloud's system. This is a number.

name
: The user's full name.

organization
: The ID of the user's active organization

organizations
: A list of IDs of all of the organizations this user is a part of.

username
: The user's username

uuid
: A unique identifier used across all of MuckRock's sites.

2.7 Organizations

Organizations using the DocumentCloud service.

2.7.1 OrganizationClient

class documentcloud.organizations.**OrganizationClient**

The organization client gives access to retrieval of organizations. It is generally accessed as `client.organizations`.

all(*self*, ***params*)

An alias for `list()`.

get(*id_*)

Return the organization with the provided DocumentCloud identifer.

list(*self*, ***params*)

Return a list of all organizations, possibly filtered by the given parameters. Please see the full API documentation for available parameters.

2.7.2 Organization

class documentcloud.organizations.**Organization**

avatar_url

A URL pointing to an image representing the organization

id

The unique identifer of the organization in DocumentCloud's system. This is a number.

individual

A boolean indicating whether this organization is for the exclusive use of an indivudal user.

name

The organization's name

slug

A URL friendly representation of the organizations's name.

uuid

A unique identifier used across all of MuckRock's sites.

2.8 Changelog

2.8.1 2.1.2

- Update base URL from api.beta.documentcloud.org to api.www.documentcloud.org

2.8.2 2.1.1

- Added original_extension field to Document
- Properly send authentication when fetching private assets

2.8.3 2.1.0

- Add a process method to document

2.8.4 2.0.2

- Add logging and error handling to upload_directory
- Add request retry logic for PUT requests to S3 and to file fetching requests
- Respect bulk limit for adding documents to a project

2.8.5 2.0.1

- Add long description to PyPI
- More complete documentation

2.8.6 2.0.0

- Initial release of re-written library for the new DocumentCloud API

2.9 Credits

The lead developer of this project is Ben Welsh.

But it would not be possible without:

- The DocumentCloud team.
- Chris Amico, Christopher Groskopf and Mitchell Kotler, who broke ground with code that I've adapted.
- Fixes from friendly people like Joe Germuska, Shane Shifflet and Adi Eyal.

CHAPTER 3

Contributing

- Code repository: https://github.com/muckrock/python-documentcloud
- Issues: https://github.com/muckrock/python-documentcloud/issues
- Packaging: https://pypi.python.org/pypi/documentcloud

Index