PyElucidate Documentation

Release 0.1.0

Matt McGrattan

Apr 18, 2019

Contents

CHAPTER 1

Installation

At the command line either via easy_install or pip:

```
$ easy_install pyelucidate
$ pip install pyelucidate
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pyelucidate
$ pip install pyelucidate
```

CHAPTER 2

Usage

To use PyElucidate in a project:

```
import pyelucidate
```

2.1 Introduction

The _PyElucidate_ library primarily consists of simple functions with no state, which interact with an instance of the Elucidate server using the W3C Web Annotation Protocol but also Elucidate's additional service APIs.

See Elucidate's usage documentation for further information.

2.2 Basic W3C Functions

2.2.1 POST a new annotation

This code will create the annotation container based on an MD5 hash of the annotation target, if a container slug is not passed in as a parameter.

The code will insert the appropriate `@context` if no `@context` is provided.

```
from pyelucidate import pyelucidate

anno = {
    "@context": "http://www.w3.org/ns/anno.jsonld",
    "type": "Annotation",
    "body": [{"value": "Foo", "purpose": "tagging"}],
    "target": {
        "type": "SpecificResource",
        "dcterms:isPartOf": {
            "id": "http://example.org/manifest/foo/manifest.json",
```

(continues on next page)

(continued from previous page)

```
            "type": "sc:Manifest",
        },
        "selector": {
            "type": "FragmentSelector",
            "conformsTo": "http://www.w3.org/TR/media-frags/",
            "value": "xywh=659,1646,174,62",
        },
        "source": "http://example.org/manifest/foo/canvas/274",
    },
    "motivation": "tagging",
}

status_code, anno_id = pyelucidate.create_anno(
    elucidate_base="https://elucidate.example.org", model="w3c", annotation=anno
)
assert status_code == 201
assert anno_id is not None
```

Post a new annotation to a specified container ("foo"):

```
from pyelucidate import pyelucidate
import requests

anno = {
    "@context": "http://www.w3.org/ns/anno.jsonld",
    "type": "Annotation",
    "body": [{"value": "Foo", "purpose": "tagging"}],
    "target": {
        "type": "SpecificResource",
        "dcterms:isPartOf": {
            "id": "http://example.org/manifest/foo/manifest.json",
            "type": "sc:Manifest",
        },
        "selector": {
            "type": "FragmentSelector",
            "conformsTo": "http://www.w3.org/TR/media-frags/",
            "value": "xywh=659,1646,174,62",
        },
        "source": "http://example.org/manifest/foo/canvas/274",
    },
    "motivation": "tagging",
}

status_code, anno_id = pyelucidate.create_anno(
    elucidate_base="https://elucidate.example.org", model="w3c", annotation=anno,
    container="foo"
)
container_contents = requests.get("https://elucidate.example.org/foo/")
assert status_code == 201
assert anno_id is not None
assert container_contents.status_code == 200
```

2.2.2 GET an annotation

Will fetch the annotation and return annotation plus the ETag for the annotation.

```
from pyelucidate import pyelucidate
import json

annotation, etag = pyelucidate.read_anno("https://elucidate.example.org/annotation/w3c
↪"
                          "/36b74ab23429078e9a8631ed4a471095/0ef3db79-c6a0-4755-a0a1-
↪8ba660f81e93")
```

2.2.3 DELETE an annotation

The W3C Web Annotation Protocol requires an `If-Match` header with the ETag for the annotation. This function requires the ETag to be provided.

If `dry_run` is `True` (the default), the function will return a 204 without deleting the annotation.

The example below shows fetching an annotation, checking the purpose for the body, and deleting the annotation.

```
from pyelucidate import pyelucidate

annotation, etag = pyelucidate.read_anno("https://elucidate.example.org/annotation/w3c
↪"
                          "/36b74ab23429078e9a8631ed4a471095/0ef3db79-c6a0-4755-a0a1-
↪8ba660f81e93")


if annotation["body"]["purpose"] == "tagging":
    status = pyelucidate.delete_anno(anno_uri = annotation["id"], etag=etag, dry_
↪run=False)
    assert status == 204
```

2.2.4 CREATE a container

It is also possible to create a container, before POSTing any annotations.

Note that *elucidate_uri* contains the full path, including the model *w3c*, and not just the base elucidate URI.

```
from pyelucidate import pyelucidate


status_code = pyelucidate.create_container(
    container_name="bar",
    label="A test container",
    elucidate_uri="https://elucidate.example.org/annotation/w3c/",
)

assert status_code in [200, 201]
```

This code will check if the container already exists, before creating, so can be run repeatedly if required.

2.2.5 PUT an updated annotation

The W3C Web Annotation Protocol requires an `If-Match` header with the ETag for the annotation. This function requires the ETag to be provided.

If `dry_run` is `True` (the default), the function will return a 200 without updating the annotation.

The example below shows fetching an annotation, updating the body, and updating the annotation.

```
from pyelucidate import pyelucidate


annotation, etag = pyelucidate.read_anno("https://elucidate.example.org/annotation/w3c
↪"
                                         "/36b74ab23429078e9a8631ed4a471095/0ef3db79-
↪c6a0-4755-a0a1-8ba660f81e93")

# Change the annotation body value
annotation["body"]["value"] = "foo"

# Put the update annotation
status_code = pyelucidate.update_anno(anno_uri=annotation["id"], anno_
↪content=annotation, etag=etag, dry_run=False)

# Check the result
annotation, etag = pyelucidate.read_anno("https://elucidate.example.org/annotation/w3c
↪"
                                         "/36b74ab23429078e9a8631ed4a471095/0ef3db79-
↪c6a0-4755-a0a1-8ba660f81e93")
assert annotation["body"]["value"] == "foo"
```

2.3 Elucidate Services

Elucidate provides a number of additional services which extend the W3C Web Annotation Protocol.

2.3.1 Query by body source

Query Elucidate for all annotations with a specified body source.

Typical usage:

> For tagging annotations, where the target is tagged with a particular topic URI, return all annotations that have been tagged with that topic URI.

For example, to find all annotations, tagged with *https://omeka.example.org/topic/virtual:person/mary+smith*:

```
from pyelucidate import pyelucidate


annotations = pyelucidate.items_by_body_source(elucidate="http://elucidate.example.org
↪",
                                               strict=True,
                                               topic="https://omeka.example.org/topic/
↪virtual:person/mary+smith")

# identifiers for annotations with body source
anno_ids = [a["id"] for a in annotations]
```

This function is a generator that yields the annotations.

The *strict* parameter sets whether Elucidate does a prefix style search or looks for an exact match.

2.3.2 Query by target

Query Elucidate for all annotations with a specified target.

Typical usage:

> For a IIIF canvas, return all annotations with that canvas as target.

PyElucidate provides a single asynchronous function (see below), but a non-asynchronous version can be done with PyElucidate's helper functions.

Search by annotation target:

```
from pyelucidate import pyelucidate
import requests
import json

# generate a search to Elucidate, using Elucidate's search API to search target id␣
↪and target source
search_uri = pyelucidate.gen_search_by_target_uri(elucidate_base="https://elucidate.
↪example.org", model="w3c",
                                                  target_uri="http://iiif.example.org/
↪iiif/manfiest/1/canvas/4",
                                                  field=["id", "source"])

r = requests.get(search_uri)

annotations = []

if r.status_code == requests.codes.ok:
    for page in pyelucidate.annotation_pages(r.json()):  # PyElucidate helper for␣
↪handling activity streams paging.
        annotations.extend(requests.get(page).json()["items"])

print(json.dumps(annotations, indent=4))
```

Search by container (assumes the container is an MD5 hash of the target URI, which is usual practice on Digirati's DLCS projects):

```
from pyelucidate import pyelucidate
import requests
import json

# generate a search to Elucidate, using Elucidate's search API to request a container
↪'s contents
search_uri = pyelucidate.gen_search_by_container_uri(elucidate_base="https://
↪elucidate.glam-dev.org", model="w3c",
                                                     target_uri="http://iiif.example.
↪org/iiif/manfiest/1/canvas/4")


r = requests.get(search_uri)

annotations = []

if r.status_code == requests.codes.ok:
    for page in pyelucidate.annotation_pages(r.json()):  # PyElucidate helper for␣
↪handling activity streams paging.
        annotations.extend(requests.get(page).json()["items"])
```

(continues on next page)

(continued from previous page)

```
print(json.dumps(annotations, indent=4))
```

2.3.3 Parents by body source

Query Elucidate for all annotations with a specified body source, and return a list of parents.

Typical usage:

> For tagging annotations, where the target is tagged with a particular topic URI, return all parent manifests for canvases that have been tagged with that topic URI.

N.B. this code sets *strict=True* on the Elucidate query.

```
from pyelucidate import pyelucidate


parents = pyelucidate.parents_by_topic(elucidate="http://elucidate.example.org",
                                       topic="https://omeka.example.org/topic/
↪virtual:person/mary+smith")


print(list(set(parents)))
```

2.3.4 Bulk update

Placeholder

2.3.5 Bulk delete

Placeholder

2.4 Asynchronous functions

Elucidate provides a number of additional services which extend the W3C Web Annotation Protocol. PyElucidate provides asynchronous versions of these functions which can make parallel requests for efficient return of results.

CHAPTER 3

pyelucidate

3.1 pyelucidate package

3.1.1 Submodules

3.1.2 pyelucidate.pyelucidate module

annotation_pages(*result: Optional[dict]*) → Optional[str]

Generator which yields URLs for annotation pages from an Activity Streams paged result set. Works by looking for the "last" page in the paged result set and incrementing between 0 and last.

Does not request each page and examine "next" or "previous".

For example, given an Activity Streams paged result set which contains:

```
{"last": "https://elucidate.example.org/annotation/w3c/services/search/body?
↪page=3&fields
=source&value=FOO&desc=1"}
```

Will yield:

https://elucidate.example.org/annotation/w3c/services/search/body?fields=source&value=FOO&desc=1&page=0

https://elucidate.example.org/annotation/w3c/services/search/body?fields=source&value=FOO&desc=1&page=1

https://elucidate.example.org/annotation/w3c/services/search/body?fields=source&value=FOO&desc=1&page=2

https://elucidate.example.org/annotation/w3c/services/search/body?fields=source&value=FOO&desc=1&page=3

Parameters **result** – Activity Streams paged result set

Returns Activity Streams page URIs.

async_items_by_container(*elucidate: str*, *container: Optional[str] = None*, *target_uri: Optional[str] = None*, *header_dict: Optional[dict] = None*, ***kwargs*) → Optional[dict]

Asynchronously yield annotations from a query by container to Elucidate.

Container can be hashed from target URI, or provided

Parameters

- **elucidate** – Elucidate server, e.g. https://elucidate.example.org
- **target_uri** – URI from target source and id, e.g. 'https://manifest.example.org/manifest/1'
- **container** – container path
- **header_dict** – dict of headers

Returns annotation object

async_items_by_creator(*elucidate: str*, *creator_id: str*, ***kwargs*) → dict

Asynchronously yield annotations from a query by creator to Elucidate.

Async requests all of the annotation pages before yielding.

Parameters

- **elucidate** – Elucidate server, e.g. https://elucidate.example.org
- **creator_id** – URI from target source and id, e.g. 'https://manifest.example.org/manifest/1'

Returns annotation object

async_items_by_target(*elucidate: str*, *target_uri: str*, ***kwargs*) → dict

Asynchronously yield annotations from a query by topic to Elucidate.

Async requests all of the annotation pages before yielding.

Parameters

- **elucidate** – Elucidate server, e.g. https://elucidate.example.org
- **target_uri** – URI from target source and id, e.g. 'https://manifest.example.org/manifest/1'

Returns annotation object

async_items_by_topic(*elucidate: str*, *topic: str*, ***kwargs*) → dict

Asynchronously yield annotations from a query by topic to Elucidate.

Does an asynchronous get for all the annotations, and then yields the annotations with optional transformation provided by the "trans_function" arg.

Parameters

- **elucidate** – Elucidate server, e.g. https://elucidate.example.org
- **topic** – URI from body source, e.g. 'https://topics.example.org/people/mary+jones'

Returns annotation object

async_manifests_by_topic(*elucidate: str*, *topic: Optional[str] = None*) → Optional[list]

Asynchronously fetch the results from a topic query to Elucidate and yield manifest URIs

N.B. assumption, if passed a string for target, rather than an object, that manifest and canvas URI patterns follow old API DLCS/Presley model.

Parameters

- **elucidate** – URL for Elucidate server, e.g. https://elucidate.example.org
- **topic** – URL for body source, e.g. https://topics.example.org/people/mary+jones

Returns manifest URI

batch_delete_target(*target_uri: str*, *elucidate_uri: str*, *dry_run: bool = True*) → int

Use Elucidate's batch delete API to delete everything with a given target id or target source URI.

https://github.com/dlcs/elucidate-server/blob/master/USAGE.md#batch-delete

Parameters

- **target_uri** – URI to delete
- **elucidate_uri** – URI of the Elucidate server, e.g. https://elucidate.example.org
- **dry_run** – if True, do not actually delete, just log request and return a 200

Returns status code

batch_delete_topic(*topic_id: str*, *elucidate_base: str*, *dry_run: bool = True*) → Tuple[int, str]

Use Elucidate's batch update apis to delete all instances of a topic URI.

https://github.com/dlcs/elucidate-server/blob/master/USAGE.md#batch-delete

Parameters

- **topic_id** – topic id to delete
- **elucidate_base** – elucidate base URI, e.g. https://elucidate.example.org
- **dry_run** – if True, will simply log and then return a 200

Returns tuple - http POST status code, JSON POSTed (as string)

batch_update_body(*new_topic_id: str*, *old_topic_ids: list*, *elucidate_base: str*, *dry_run: bool = True*) → Tuple[int, dict]

Use Elucidate's bulk update APIs to replace all instances of each of a list of body source or id URIs (aka a topic) with the new URI (aka topic).

https://github.com/dlcs/elucidate-server/blob/master/USAGE.md#batch-update

Parameters

- **new_topic_id** – topic ids to use, string
- **old_topic_ids** – topic ids to replace, list
- **elucidate_base** – elucidate base URI, e.g. https://elucidate.example.org
- **dry_run** – if True, will simply log JSON and URI and then return a 200

Returns POST status code

create_anno(*elucidate_base: str*, *annotation: dict*, *target: Optional[str] = None*, *container: Optional[str] = None*, *model: Optional[str] = 'w3c'*) → Tuple[int, Optional[str]]

POST an annotation to Elucidate, can be optionally passed a container, if container is None will use the MD5 hash of the manifest or canvas target URI as the container name.

If no @context is provided, the code will insert the appropriate context based on the model.

Parameters

- **elucidate_base** – base URI for the annotation server, e.g. https://elucidate.example.org

- **target** – target for the annotation (optional), will attempt to parse anno for target if not present
- **annotation** – annotation object
- **container** – container name (optional), will use hash of target uri if not present
- **model** – oa or w3c

Returns status code from Elucidate, annotation id (or none)

create_container(*container_name: str*, *label: str*, *elucidate_uri: str*) → int
Create an annotation container with a container name and label.

Parameters

- **container_name** – name of the container
- **label** – label for the container
- **elucidate_uri** – uri for the annotation server, including full path, e.g. https://elucidate.example.org/annotation/w3c/

Returns POST request status code

delete_anno(*anno_uri: str*, *etag: str*, *dry_run: bool = True*) → int
Delete an individual annotation, requires etag.

Optionally, can be run as a dry run which will not delete the annotation.

Parameters

- **anno_uri** – URI for annotation
- **etag** – ETag
- **dry_run** – if True, log and return a 204

Returns return DELETE request status code

fetch(*url: str*, *session: aiohttp.client.ClientSession*) → dict
Asynchronously fetch a url, using specified ClientSession.

fetch_all(*urls: list*, *connector_limit: int = 5*) → _asyncio.Future
Launch async requests for all web pages in list of urls.

Parameters

- **urls** – list of URLs to fetch
- **connector_limit** – integer for max parallel connections

:return results from requests

format_results(*annotation_list: Optional[list], request_uri: str*) → Optional[dict]
Takes a list of annotations and returns as a standard Presentation API Annotation List.

Parameters

- **annotation_list** – list of annotations
- **request_uri** – the URI to use for the @id

:return dict or None

gen_search_by_container_uri(*elucidate_base: str, target_uri: Optional[str], model: str = 'w3c'*) → Optional[str]
Return the annotation container uri for a target. Assumes that the container URI is an md5 hash of the target URI (as per current DLCS general practice).

This URI can be passed to other functions to return the result of the query.

Parameters

- **elucidate_base** – base URI for the annotation server, e.g. https://elucidate.example.org
- **target_uri** – target URI to search for, e.g. IIIF Presentation API manifest or canvas URI
- **model** – oa or w3c

Returns uri

gen_search_by_target_uri(*target_uri: Optional[str], elucidate_base: str, model: str = 'w3c', field=None*) → Optional[str]

Returns a search URI for searching Elucidate for a target using Elucidate's basic search API.

This URI can be passed to other functions to return the result of the query.

Parameters

- **model** – oa or w3c, defaults to w3c.
- **elucidate_base** – base URI for the annotation server, e.g. https://elucidate.example.org
- **target_uri** – target URI to search for, e.g. a IIIF Presentatiion API canvas or manifest

URI :param field: list of fields to search on, defaults to both source and id :return: uri

get_items(*uri: str*) → Optional[dict]

Page through an ActivityStreams paged result set, yielding each page's items one at a time.

Parameters **uri** – Request URI, e.g. provided by gen_search_by_target_uri()

Returns item

identify_target(*annotation_content: dict*) → Optional[str]

Identify the base level target for an annotation, for

https://example.org/foo#XYWH=0,0,200,200

output

https://example.org/foo

If the annotation has multiple targets, return just base level target for the first.

Parameters **annotation_content** – annotation dict

Returns uri

iiif_batch_delete_by_manifest(*manifest_uri: str, elucidate_uri: str, dry_run: bool = True*) → bool

Provides a IIIF aware wrapper around the _batch_delete_by_target_ function. Requests a IIIF Presentation API manifest and deletes all of the annotations with the canvas or the manifest URIs as their target.

Use Elucidate's batch delete API to delete everything with a given target id or target source URI.

https://github.com/dlcs/elucidate-server/blob/master/USAGE.md#batch-delete

Parameters

- **manifest_uri** – URI of IIIF Presentation API manifest (must be de-referenceable)
- **elucidate_uri** – base URI for Elucidate, e.g. https://elucidate.example.org
- **dry_run** – if True, will not actually delete the content

Returns boolean for status, True if no errors, False if error on any delete operation.

iiif_iterative_delete_by_manifest (*manifest_uri: str*, *elucidate_uri: str*, *method: str = 'search'*, *dry_run: bool = True*) → bool

Provides a IIIF aware wrapper around the iterative_delete_by_target function.

Iteratively delete all annotations for every canvas in a IIIF Presentation manifest and for the IIIF Presentation API manifest itself.

Requests annotations either by container or by target URI and iteratively deletes the annotations by id, one at a time, using HTTP DELETE.

Does not use Elucidate's batch delete APIs.

Parameters

- **dry_run** – if True, will not actually delete
- **method** – identify the annotations to delete via container (hash) or search (Elucidate

query) :param manifest_uri: URI for IIIF Presentation API manifest. :param elucidate_uri: Elucidate base URI, e.g. https://elucidate.example.org :return: boolean success or fail

iiif_iterative_delete_by_manifest_async_get (*manifest_uri: str*, *elucidate_uri: str*, *dry_run: bool = True*) → bool

Delete all annotations for every canvas in a IIIF manifest and for the manifest.

Uses asynchronous code to parallel get the search results to build the annotation list.

N.B. does NOT do an async DELETE. Delete is sequential.

Parameters

- **dry_run** – if True, will not actually delete, just prints URIs
- **manifest_uri** – uri for IIIF manifest
- **elucidate_uri** – Elucidate base uri

Returns boolean success or fail

item_ids (*item: dict*) → Optional[str]

Small helper function to yield identifier URI(s) for item from an Activity Streams item. Will yield both '@id' and 'id' values.

Parameters **item** – Item from an activity streams page

Returns uri

items_by_body_source (*elucidate: str*, *topic: str*, *strict: bool = True*) → dict

Generator to yield annotations from query to Elucidate by body source.

For example, for a W3C web annotation, with body:

```
{"body": [
          {
            "type": "SpecificResource",
            "format": "application/html",
            "creator": "https://montague.example.org/",
            "generator": "https://montague.example.org//nlp/",
            "purpose": "tagging",
            "source": "https://www.example.org/themes/foo"
          }
      ]}
```

This function will query Elucidate for all annotations with body id or body source == "https://www.example.org/themes/foo".

If strict = False, this would match both:

https://www.example.org/themes/foo

and

https://www.example.org/themes/foobar

If strict = True, only annotations with an exact match on the body source will be returned.

Parameters

- **elucidate** – URL for Elucidate server, e.g. https://elucidate.example.org
- **topic** – URI for body source, e.g. https://www.example.org/themes/foo
- **strict** – if strict, use strict = True.

Returns annotation dict

iterative_delete_by_target(*target: str*, *elucidate_base: str*, *search_method: str = 'container'*, *dryrun: bool = True*) → bool

Delete all annotations in a container for a target URI. Works by querying for the annotations and then iteratively deleting them one at a time.

Note, that this is _not_ an operation using Elucidate's batch delete APIs.

Negative: could be slow, and involve many consecutive HTTP requests

Positive: as the code is handling the annotations one at a time, it will not time out with very large result sets.

The function can build the list of annotations to delete using either:

the Elucidate search by target API,

or a hash of the target URI to get a container URI.

N.B. choosing the container method assumes that container ID as an MD5 hash of the target URI.

Parameters

- **dryrun** – if True, will not actually delete, just logs and returns True (for success)
- **search_method** – 'container' (hash of target URI) or 'search' (Elucidate query by target)
- **target** – target URI
- **elucidate_base** – base URI for Elucidate, e.g. https://elucidate.example.org

Returns boolean success or fail, True if no errors on _any_ request.

iterative_delete_by_target_async_get(*target: str*, *elucidate_base: str*, *dryrun: bool = True*) → bool

Delete all annotations in a container for a target uri. Works by querying for the annotations and then iteratively deleting them one at a time. Not a bulk delete operation using Elucidate's bulk APIs.

N.B. Negative: could be slow, and involve many HTTP requests, Positive: doesn't really matter how big the result set is, it won't time out, as handling the annotations one at a time.

Asynchronous query using the Elucidate search by target API to fetch the list of annotations to delete.

DELETE is not asychronous, but sequential.

Parameters

- **dryrun** – if True, will not actually delete, just logs and returns True (for success)
- **target** – target uri
- **elucidate_base** – base URI for Elucidate, e.g. https://elucidate.example.org

Returns boolean success or fail, True if no errors on _any_ request.

mirador_oa (*w3c_body: dict*) → dict
Transform a single W3C Web Annotation Body (e.g. as produced by Montague) and returns formatted for Open Annotation in the Mirador client.

Parameters **w3c_body** – annotation body

Returns transformed annotation body

parent_from_annotation (*content: dict*) → Optional[str]
Parse W3C web annotation and attempt to yield URI for parent object the annotation target is part of.

A typical use would be to return the parent IIIF Presentation API manifest URI for an annotation on a IIIF Presentation API canvas or fragment of a canvas.

The code makes the assumption that, if passed a string for target, rather than an object, that manifest and canvas URI patterns follow the model used by the RESTful DLCS API model.

On this pattern, a canvas with URI:

https://example.org/iiif/foo/canvas/c1

will have a parent manifest with URI:

https://example.org/iiif/foo/manifest

This assumption may not, and probably will not, hold for other sources.

If the annotation has a "dcterms:isPartOf" field within the target, the value of "dcterms:isPartOf" will be returned. If there are a list of annotation targets, the first parent will be returned.

Parameters **content** – annotation object

Returns target parent URI

parents_by_topic (*elucidate: str*, *topic: str*) → Optional[str]
Generator parses results from an Elucidate topic search request, and yields parent/manifest URIs.

The code makes the assumption that, if passed a string for target, rather than an object, that manifest and canvas URI patterns follow the model used by the RESTful DLCS API model.

On this pattern, a canvas with URI:

https://example.org/iiif/foo/canvas/c1

will have a parent manifest with URI:

https://example.org/iiif/foo/manifest

This assumption may not, and probably will not, hold for other sources.

Parameters

- **elucidate** – URL for Elucidate server, e.g. https://elucidate.example.org
- **topic** – URL for body source, e.g. https://topics.example.org/people/mary+jones

Returns manifest URI

read_anno (*anno_uri: str) -> (typing.Union[str, NoneType], typing.Union[str, NoneType]*)
GET an annotation from Elucidate, returns a tuple of annotation content and ETag

Parameters **anno_uri** – URI for annotation

Returns annotation content, etag

remove_keys(*d: dict*, *keys: list*) → dict
Remove keys from a dictionary.

Parameters

- **d** – dict to edit
- **keys** – list of keys to remove

Returns dict with keys removed

set_query_field(*url: str, field: str, value: Union[int, str], replace: bool = False*)
Parse out the different parts of a URL, and optionally replace a query string parameter, before return the unparsed new URL.

Parameters

- **url** – URL to parse
- **field** – field where the value should be replaced
- **value** – replacement value
- **replace** – boolean, if True, replace query string parameter

Returns unparsed URL

target_extract(*json_dict: dict*, *fake_selector: bool = False*) → Optional[str]
Extract the target and turn into a simple 'on'.

Optionally, fake a selector, e.g. for whole canvas annotations, generate a target XYWH bounding box at top left.

Parameters

- **fake_selector** – if True, create a top left 50px box and associate with that.
- **json_dict** – annotation content as dictionary

Returns string for the target URI

transform_annotation(*item: dict*, *flatten_at_ids: bool = True*, *transform_function: Optional[Callable] = None*) → Optional[dict]
Transform an annotation given an arbitrary function that is passed in.

For example, W3C to OA using "mirador_oa".

The function will remove keys not used in the Open Annotation model.

If no transform_function is provided the annotation will be returned unaltered.

Parameters

- **item** – annotation
- **flatten_at_ids** – if True replace @id dict with simple "@id" : "foo"
- **transform_function** – function to pass the annotation through

Returns

update_anno(*anno_uri: str*, *anno_content: dict*, *etag: str*, *dry_run: bool = True*) → int
Update an individual annotation, requires etag.

Optionally, can be run as a dry run which will not update the annotation but will return a 200.

Parameters

- **anno_uri** – URI for annotation
- **anno_content** – the annotation content

- **etag** – ETag
- **dry_run** – if True, log and return a 200

Returns return PUT request status code

uri_contract(*uri: str*) → Optional[str]

Contract a URI to just the schema, netloc, and path

For example, for:

https://example.org/foo#XYWH=0,0,200,200

Returns //example.org/foo

Return type https

Parameters **uri** – URI to contract

Returns contracted URI

3.1.3 Module contents

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated!

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/digirati-co-uk/pyelucidate/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” is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/digirati-co-uk/pyelucidate/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.

4.2 Get Started!

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

1. Fork the *pyelucidate* repo on GitHub.
2. Clone your fork locally:

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

3. Create a branch for local development:

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

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

```
$ tox
```

To get tox, just pip install it.

5. 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
```

6. 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 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/digirati-co-uk/pyelucidate under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test test/test_pyelucidate.py
```

CHAPTER 5

Credits

5.1 Development Lead

- Matt McGrattan <matt.mcgrattan@digirati.com>

5.2 Contributors

CHAPTER 6

History

6.1 0.1.0 (2018-10-12)

- First release on PyPI.

CHAPTER 7

License

CHAPTER 8

PyElucidate

Open Source Python Tools for the Elucidate Annotation Server.

8.1 Introduction

Simple Python helpers for working with Elucidate, an Open Annotation and W3C Web Annotation annotation server.

Elucidate uses the W3C Web Annotation Protocol but also has some useful additional service APIs. See Elucidate's usage documentation for further information.

Full documentation at Readthedocs.

8.2 Requirements

Python 3.5+

Required python packages are listed in *requirements.txt*.

CHAPTER 9

Installation

At the command line either via easy_install or pip:

```
$ easy_install pyelucidate
$ pip install pyelucidate
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pyelucidate
$ pip install pyelucidate
```

9.1 Usage

Placeholder

PyElucidate uses Python 3.x. No backwards compatibility with Python 2.x is provided.

Feel free to raise Github issues.

If you find an issue you are interested in fixing you can:

- make sure the issue is small and atomic
- Fork the repository
- Clone the repository to your local machine
- Create a new branch for your fix using *git checkout -b branch-name-here*.
- Fix the issue.
- Commit and push the code to your remote repository.
- Submit a pull request to the *pyelucidate* repository, with a description of your fix and the issue number.
- The PR will be reviewed by the maintainer and either merge the PR or response with comments.

9.2 License

CHAPTER 10

Feedback

If you have any suggestions or questions about **PyElucidate** feel free to email me at matt.mcgrattan@digirati.com.

If you encounter any errors or problems with **PyElucidate**, please let me know! Open an Issue at the GitHub http://github.com/digirati-co-uk/pyelucidate main repository.

Python Module Index

p

Index