Rauth Documentation

Release 0.7.3

Max Countryman

Sep 27, 2017

Contents

1 Installation **3**

2 Usage **5**

3 API Reference **7**
3.1 API 7

4 Upgrade Notes **25**
4.1 Upgrading Rauth 25
4.2 Rauth Changelog 26

Python Module Index **29**

A simple Python OAuth 1.0/a, OAuth 2.0, and Ofly consumer library built on top of Requests.

CHAPTER 1

Installation

Install the module with one of the following commands:

```
$ pip install rauth
```

Or if you must:

```
$ easy_install rauth
```

CHAPTER 2

Usage

If you want to check out the complete *API* documentation, go ahead.

The easiest way to get started is by setting up a service wrapper. To do so simply import the service container object:

```
from rauth import OAuth2Service

facebook = OAuth2Service(
    client_id='440483442642551',
    client_secret='cd54f1ace848fa2a7ac89a31ed9c1b61',
    name='facebook',
    authorize_url='https://graph.facebook.com/oauth/authorize',
    access_token_url='https://graph.facebook.com/oauth/access_token',
    base_url='https://graph.facebook.com/')
```

Using the service wrapper API we can obtain an access token after the authorization URL has been visited by the client. First generate the authorization URL:

```
redirect_uri = 'https://www.facebook.com/connect/login_success.html'
params = {'scope': 'read_stream',
          'response_type': 'code',
          'redirect_uri': redirect_uri}

url = facebook.get_authorize_url(**params)
```

Once this URL has been visited and (presumably) the client authorizes the application an access token can be obtained:

```
# the code should be returned upon the redirect from the authorize step,
# be sure to use it here (hint: it's in the URL!)
session = facebook.get_auth_session(data={'code': 'foo',
                                          'redirect_uri': redirect_uri})

print session.get('me').json()['username']
```

Here is an example using the OAuth 1.0/a service wrapper:

```
from rauth import OAuth1Service

twitter = OAuth1Service(
    consumer_key='J8MoJG4bQ9gcmGh8H7XhMg',
    consumer_secret='7WAscbSy65GmiVOvMU5EBYn5z80fhQkcFWSLMJJu4',
    name='twitter',
    access_token_url='https://api.twitter.com/oauth/access_token',
    authorize_url='https://api.twitter.com/oauth/authorize',
    request_token_url='https://api.twitter.com/oauth/request_token',
    base_url='https://api.twitter.com/1/')
```

Now it's possible to obtain request tokens via *request_token = twitter.get_request_token()*, generate authorization URIs *twitter.get_authorize_url(request_token)*, and finally obtain an authenticated session *twitter.get_auth_session(request_token, request_token_secret)*.

CHAPTER 3

API Reference

Information regarding the consumer API.

API

The API is exposed via service wrappers, which provide convenient OAuth 1.0, 2.0, and Ofly flow methods as well as session management.

Each service type has specialized Session objects, which may be used directly.

OAuth 1.0 Services

class `rauth.`**`OAuth1Service`**(*consumer_key*, *consumer_secret*, *name=None*, *request_token_url=None*, *access_token_url=None*, *authorize_url=None*, *base_url=None*, *session_obj=None*, *signature_obj=None*)

An OAuth 1.0/a Service container.

This class provides a wrapper around a specialized Requests' `Session` object. Primarily this wrapper is used to produce authenticated session objects. These may be used to make requests against OAuth 1.0/a endpoints.

You might intialize *`OAuth1Service`* something like this:

```
service = OAuth1Service(
           name='example',
           consumer_key='123',
           consumer_secret='456',
           request_token_url='http://example.com/request_token',
           access_token_url='http://example.com/access_token',
           authorize_url='http://example.com/authorize',
           base_url='http://example.com/api')
```

Now the request token should be retrieved:

```
request_token, request_token_secret = service.get_request_token()
```

Differing Request Token Formats

Some services provide different formatting when returning tokens. For this reason the service wrapper provides a special method *get_raw_request_token()*. This will return the unparsed response. At this point it's up to you to extract the necessary data.

It's time to access the authorize URI and direct the client to authorize requests on their behalf. This URI is retrieved as follows:

```
authorize_url = service.get_authorize_url(request_token)
```

Once the client has authorized the request it is now possible to retrieve an access token. Do so as follows:

```
session = service.get_auth_session(request_token, request_token_secret)
```

Differing Access Token Formats

Some services provide different formatting when returning tokens. For this reason the service wrapper provides a special method *get_raw_access_token()*. This will return the unparsed response. At this point it's up to you to extract the necessary data.

Finally we have an authenticated session and are ready to make requests against OAuth 1.0/a endpoints. Because Rauth is a wrapper around Requests, the same API you would use with Requests is exposed and expected:

```
r = session.get('some/resource/', params={'format': 'json'})
print r.json()
```

Parameters

- **consumer_key** (*str*) – Client consumer key, required for signing.
- **consumer_secret** (*str*) – Client consumer secret, required for signing.
- **name** (*str*) – The service name, defaults to *None*.
- **request_token_url** (*str*) – Request token endpoint, defaults to *None*.
- **access_token_url** (*str*) – Access token endpoint, defaults to *None*.
- **authorize_url** (*str*) – Authorize endpoint, defaults to *None*.
- **base_url** (*str*) – A base URL from which to construct requests, defaults to *None*.
- **session_obj** (Session) – Object used to construct sessions with, defaults to *rauth.OAuth1Session*
- **signature_obj** (SignatureMethod) – Object used to construct signatures with, defaults to `rauth.oauth.HmacSha1Signature`

consumer_key = None
Client credentials.

get_access_token(*request_token*, *request_token_secret*, *method='GET'*, *decoder=<function parse_utf8_qsl>*, *key_token='oauth_token'*, *key_token_secret='oauth_token_secret'*, ***kwargs*)
Returns an access token pair.

Parameters

- **request_token** (*str*) – The request token as returned by *get_request_token()*.
- **request_token_secret** (*str*) – The request token secret as returned by *get_request_token()*.
- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *GET*.
- **decoder** (*func*) – A function used to parse the Response content. Should return a dictionary.
- **key_token** – The key the access token will be decoded by, defaults to 'oauth_token'.
- **key_token_secret** – The key the access token will be decoded by, defaults to 'oauth_token_secret'.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_auth_session(*request_token*, *request_token_secret*, *method='GET'*, ***kwargs*)
Gets an access token, intializes a new authenticated session with the access token. Returns an instance of *session_obj*.

Parameters

- **request_token** (*str*) – The request token as returned by *get_request_token()*.
- **request_token_secret** (*str*) – The request token secret as returned by *get_request_token()*.
- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *GET*.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_authorize_url(*request_token*, ***params*)
Returns a formatted authorize URL.

Parameters

- **request_token** (*str*) – The request token as returned by *get_request_token*.
- ****params** (*dict*) – Additional keyworded arguments to be added to the request querystring.

get_raw_access_token(*request_token*, *request_token_secret*, *method='GET'*, ***kwargs*)
Returns a Requests' response over the `rauth.OAuth1Service.access_token_url`.

Use this if your endpoint if you need the full *Response* object.

Parameters

- **request_token** (*str*) – The request token as returned by *get_request_token()*.
- **request_token_secret** (*str*) – The request token secret as returned by *get_request_token()*.
- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *GET*.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_raw_request_token(*method='GET'*, ***kwargs*)
Returns a Requests' response over the *rauth.OAuth1Service.request_token_url*.

Use this if your endpoint if you need the full *Response* object.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *GET*.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_request_token(*method='GET'*, *decoder=<function parse_utf8_qsl>*, *key_token='oauth_token'*, *key_token_secret='oauth_token_secret'*, ***kwargs*)
Return a request token pair.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *GET*.
- **decoder** (*func*) – A function used to parse the Response content. Should return a dictionary.
- **key_token** – The key the access token will be decoded by, defaults to 'oauth_token'.
- **key_token_secret** – The key the access token will be decoded by, defaults to 'oauth_token_secret'.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_session(*token=None*, *signature=None*)
If provided a *token* parameter, tries to retrieve a stored *rauth.OAuth1Session* instance. Otherwise generates a new session instance with the *rauth.OAuth1Service.consumer_key* and `rauth.OAuth1Service.consumer_secret` stored on the *rauth.OAuth1Service* instance.

Parameters **token** (*tuple*) – A tuple of strings with which to memoize the session object instance.

request_token_response = None
Request and access token responses.

request_token_url = None
Authorization endpoints.

session_obj = None
Object used to construct sessions with.

signature_obj = None
Object used to construct signatures with.

OAuth 2.0 Services

class `rauth.`**OAuth2Service**(*client_id*, *client_secret*, *name=None*, *access_token_url=None*, *authorize_url=None*, *base_url=None*, *session_obj=None*)
An OAuth 2.0 Service container.

This class provides a wrapper around a specialized Requests' `Session` object. Primarily this wrapper is used for producing authenticated session objects which are used to make requests against OAuth 2.0 endpoints.

You might intialize *OAuth2Service* something like this:

```
service = OAuth2Service(
           name='example',
           client_id='123',
           client_secret='456',
           access_token_url='https://example.com/token',
           authorize_url='https://example.com/authorize',
           base_url='https://example.com/api/')
```

Given the simplicity of OAuth 2.0 now this object *service* can be used to retrieve an authenticated session in two simple steps:

```
# the return URL is used to validate the request
params = {'redirect_uri': 'http://example.com/',
          'response_type': 'code'}
url = service.get_authorize_url(**params)

# once the above URL is consumed by a client we can ask for an access
# token. note that the code is retrieved from the redirect URL above,
# as set by the provider
data = {'code': 'foobar',
        'grant_type': 'authorization_code',
        'redirect_uri': 'http://example.com/'}

session = service.get_auth_session(data=data)
```

Now that we have retrieved a session, we may make requests against the OAuth 2.0 provider's endpoints. As much as possible the Requests' API is preserved and you may make requests using the same parameters you would using Requests:

```
r = session.get('foo', params={'format': 'json'})
print r.json()
```

Parameters

- **client_id** (*str*) – Client id.
- **client_secret** (*str*) – Client secret.
- **name** (*str*) – The service name, defaults to *None*.
- **access_token_url** (*str*) – Access token endpoint, defaults to *None*.
- **authorize_url** (*str*) – Authorize endpoint, defaults to *None*.
- **base_url** (*str*) – A base URL from which to construct requests, defaults to *None*.
- **session_obj** (`rauth.Session`) – Object used to construct sessions with, defaults to *OAuth2Session*

access_token_response = None
Access token response.

access_token_url = None
The provider's access token URL.

client_id = None
Client credentials.

get_access_token(*method='POST'*, *decoder=<function parse_utf8_qsl>*, *key='access_token'*, ***kwargs*)
Returns an access token.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *POST*.
- **decoder** (*func*) – A function used to parse the Response content. Should return a dictionary.
- **key** – The key the access token will be decoded by, defaults to 'access_token'.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_auth_session(*method='POST'*, ***kwargs*)
Gets an access token, intializes a new authenticated session with the access token. Returns an instance of *session_obj*.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *POST*.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_authorize_url(***params*)
Returns a formatted authorize URL.

Parameters ****params** (*dict*) – Additional keyworded arguments to be added to the URL querystring.

get_raw_access_token(*method='POST'*, ***kwargs*)
Returns a Requests' response over the *OAuth2Service.access_token_url*.

Use this if your endpoint if you need the full *Response* object.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used, defaults to *POST*.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_session(*token=None*)
If provided, the *token* parameter is used to initialize an authenticated session, otherwise an unauthenticated session object is generated. Returns an instance of *session_obj*..

Parameters **token** (*str*) – A token with which to initilize the session.

session_obj = None
Object used to construct sessions with.

Ofly Services

class rauth.**OflyService**(*app_id*, *app_secret*, *name=None*, *authorize_url=None*, *base_url=None*, *user_id=None*, *session_obj=None*)
An Ofly Service container.

This class wraps an Ofly service i.e., Shutterfly. The process is similar to that of OAuth 1.0 but simplified.

You might intialize *OflyService* something like this:

```
service = OflyService(name='example',
                      app_id='123',
                      app_secret='456',
                      authorize_url='http://example.com/authorize')
```

A signed authorize URL is then produced via calling *service.get_authorize_url*. Once this has been visited by the client and assuming the client authorizes the request.

Normal API calls can now be made using a session instance. Retrieve the authenticated session like so:

```
session = service.get_auth_session('foo')

# now we can make regular Requests' calls
r = session.get('bar')
```

Parameters

- **app_id** (*str*) – The oFlyAppId, i.e. "application ID".
- **app_secret** (*str*) – The oFlyAppSecret, i.e. "shared secret".
- **name** (*str*) – The service name, defaults to *None*.
- **authorize_url** (*str*) – Authorize endpoint, defaults to *None*.
- **base_url** (*str*) – A base URL from which to construct requests, defaults to *None*.
- **user_id** (*str*) – The oflyUserid, defaults to *None*. Note: this is required for Ofly requests, retrieved via authorize URL.
- **session_obj** (`rauth.Session`) – Object used to construct sessions with, defaults to *rauth.OflySession*

app_id = None
Client credentials.

get_auth_session(*user_id*, ***kwargs*)
Intializes a new authenticated session with *user_id* as oFlyUserid. Returns an instance of *session_obj*.

Parameters

- **user_id** (*str*) – The oflyUserid, defaults to *None*.
- ****kwargs** (*dict*) – Optional arguments. Same as Requests.

get_authorize_url(***params*)
Returns a formatted authorize URL.

Parameters ****params** (*dict*) – Additional keyworded arguments to be added to the request querystring.

get_session(*token*)
The token parameter should be *oFlyUserid*. This is used to initialize an authenticated session instance. Returns an instance of *session_obj*.

Parameters **token** (*str*) – A token with which to initialize the session with, e.g. *OflyService.user_id*.

session_obj = None
Object used to construct sessions with.

user_id = None
The oflyUserid.

OAuth 1.0 Sessions

class rauth.**OAuth1Session**(*consumer_key*, *consumer_secret*, *access_token=None*, *access_token_secret=None*, *signature=None*, *service=None*)

A specialized Session object, wrapping OAuth 1.0/a logic.

This object is utilized by the *OAuth1Service* wrapper but can be used independently of that infrastructure. Essentially this is a loose wrapping around the standard Requests codepath. State may be tracked at this layer, especially if the instance is kept around and tracked via some unique identifier, e.g. access tokens. Things like request cookies will be preserved between requests and in fact all functionality provided by a Requests' Session object should be exposed here.

If you were to use this object by itself you could do so by instantiating it like this:

```
session = OAuth1Session('123',
                        '456',
                        access_token='321',
                        access_token_secret='654')
```

You now have a session object which can be used to make requests exactly as you would with a normal Requests' Session instance. This anticipates that the standard OAuth 1.0/a flow will be modeled outside of the scope of this class. In other words, if the fully qualified flow is useful to you then this object probably need not be used directly, instead consider using *OAuth1Service*.

Once the session object is setup, you may start making requests:

```
r = session.get('http://example/com/api/resource',
                params={'format': 'json'})
print r.json()
```

Parameters

- **consumer_key** (*str*) – Client consumer key.
- **consumer_secret** (*str*) – Client consumer secret.
- **access_token** (*str*) – Access token, defaults to *None*.
- **access_token_secret** (*str*) – Access token secret, defaults to *None*.
- **signature** (rauth.oauth.Signature) – A signature producing object, defaults to rauth.oauth.HmacSha1Signature.
- **service** (rauth.Service) – A back reference to the service wrapper, defaults to *None*.

access_token = None
Access token credentials.

close()
Closes all adapters and as such the session

consumer_key = None
Client credentials.

delete(*url*, ***kwargs*)
Sends a DELETE request. Returns Response object.

Parameters

- **url** – URL for the new Request object.
- ****kwargs** – Optional arguments that request takes.

Return type requests.Response

get(*url*, ***kwargs*)
Sends a GET request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

get_adapter(*url*)
Returns the appropriate connection adapter for the given URL.

Return type requests.adapters.BaseAdapter

get_redirect_target(*resp*)
Receives a Response. Returns a redirect URI or `None`

head(*url*, ***kwargs*)
Sends a HEAD request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

merge_environment_settings(*url*, *proxies*, *stream*, *verify*, *cert*)
Check the environment and merge it with some settings.

Return type dict

mount(*prefix*, *adapter*)
Registers a connection adapter to a prefix.

Adapters are sorted in descending order by prefix length.

options(*url*, ***kwargs*)
Sends a OPTIONS request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

patch(*url*, *data=None*, ***kwargs*)
Sends a PATCH request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

post (*url*, *data=None*, *json=None*, ***kwargs*)
Sends a POST request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- **json** – (optional) json to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

prepare_request (*request*)
Constructs a `PreparedRequest` for transmission and returns it. The `PreparedRequest` has settings merged from the `Request` instance and those of the `Session`.

Parameters **request** – `Request` instance to prepare with this session's settings.

Return type requests.PreparedRequest

put (*url*, *data=None*, ***kwargs*)
Sends a PUT request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

rebuild_auth (*prepared_request*, *response*)
When being redirected we may want to strip authentication from the request to avoid leaking credentials. This method intelligently removes and reapplies authentication where possible to avoid credential loss.

rebuild_method (*prepared_request*, *response*)
When being redirected we may want to change the method of the request based on certain specs or browser behavior.

rebuild_proxies (*prepared_request*, *proxies*)
This method re-evaluates the proxy configuration by considering the environment variables. If we are redirected to a URL covered by NO_PROXY, we strip the proxy configuration. Otherwise, we set missing proxy keys for this URL (in case they were stripped by a previous redirect).

This method also replaces the Proxy-Authorization header where necessary.

Return type dict

request (*method*, *url*, *header_auth=False*, *realm=''*, ***req_kwargs*)
A loose wrapper around Requests' `Session` which injects OAuth 1.0/a parameters.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used.
- **url** (*str*) – The resource to be requested.
- **header_auth** (*bool*) – Authentication via header, defaults to *False.*
- **realm** (*str*) – The auth header realm, defaults to `""`.

- ****req_kwargs** (*dict*) – Keyworded args to be passed down to Requests.

resolve_redirects(*resp*, *req*, *stream=False*, *timeout=None*, *verify=True*, *cert=None*, *proxies=None*, *yield_requests=False*, ***adapter_kwargs*)
Receives a Response. Returns a generator of Responses or Requests.

send(*request*, ***kwargs*)
Send a given PreparedRequest.

Return type requests.Response

signature = None
Signing method.

OAuth 2.0 Sessions

class rauth.**OAuth2Session**(*client_id=None*, *client_secret=None*, *access_token=None*, *service=None*, *access_token_key=None*)
A specialized Session object, wrapping OAuth 2.0 logic.

This object is utilized by the *OAuth2Service* wrapper but can be used independently of that infrastructure. Essentially this is a loose wrapping around the standard Requests codepath. State may be tracked at this layer, especially if the instance is kept around and tracked via some unique identifier, e.g. access token. Things like request cookies will be preserved between requests and in fact all functionality provided by a Requests' Session object should be exposed here.

If you were to use this object by itself you could do so by instantiating it like this:

```
session = OAuth2Session('123', '456', access_token='321')
```

You now have a session object which can be used to make requests exactly as you would with a normal Requests Session instance. This anticipates that the standard OAuth 2.0 flow will be modeled outside of the scope of this class. In other words, if the fully qualified flow is useful to you then this object probably need not be used directly, instead consider using *OAuth2Service*.

Once the session object is setup, you may start making requests:

```
r = session.get('https://example/com/api/resource',
                params={'format': 'json'})
print r.json()
```

Parameters

- **client_id** (*str*) – Client id, defaults to *None*.
- **client_secret** (*str*) – Client secret, defaults to *None*
- **access_token** (*str*) – Access token, defaults to *None*.
- **access_token_key** (*str*) – The name of the access token key, defaults to *'access_token'*.
- **service** (rauth.Service) – A back reference to the service wrapper, defaults to *None*.
- **access_token_key** – The name of the access token key, defaults to *'access_token'*.

access_token = None
Access token.

access_token_key = None
Access token key, e.g. 'access_token'.

client_id = None
Client credentials.

close()
Closes all adapters and as such the session

delete(*url*, ***kwargs*)
Sends a DELETE request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

get(*url*, ***kwargs*)
Sends a GET request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

get_adapter(*url*)
Returns the appropriate connection adapter for the given URL.

Return type requests.adapters.BaseAdapter

get_redirect_target(*resp*)
Receives a Response. Returns a redirect URI or `None`

head(*url*, ***kwargs*)
Sends a HEAD request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

merge_environment_settings(*url*, *proxies*, *stream*, *verify*, *cert*)
Check the environment and merge it with some settings.

Return type dict

mount(*prefix*, *adapter*)
Registers a connection adapter to a prefix.

Adapters are sorted in descending order by prefix length.

options(*url*, ***kwargs*)
Sends a OPTIONS request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

patch(*url*, *data=None*, ***kwargs*)
Sends a PATCH request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

post(*url*, *data=None*, *json=None*, ***kwargs*)
Sends a POST request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- **json** – (optional) json to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

prepare_request(*request*)
Constructs a `PreparedRequest` for transmission and returns it. The `PreparedRequest` has settings merged from the `Request` instance and those of the `Session`.

Parameters **request** – `Request` instance to prepare with this session's settings.

Return type requests.PreparedRequest

put(*url*, *data=None*, ***kwargs*)
Sends a PUT request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

rebuild_auth(*prepared_request*, *response*)
When being redirected we may want to strip authentication from the request to avoid leaking credentials. This method intelligently removes and reapplies authentication where possible to avoid credential loss.

rebuild_method(*prepared_request*, *response*)
When being redirected we may want to change the method of the request based on certain specs or browser behavior.

rebuild_proxies(*prepared_request*, *proxies*)
This method re-evaluates the proxy configuration by considering the environment variables. If we are redirected to a URL covered by NO_PROXY, we strip the proxy configuration. Otherwise, we set missing proxy keys for this URL (in case they were stripped by a previous redirect).

This method also replaces the Proxy-Authorization header where necessary.

Return type dict

request(*method*, *url*, *bearer_auth=True*, ***req_kwargs*)
A loose wrapper around Requests' `Session` which injects OAuth 2.0 parameters.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used.
- **url** (*str*) – The resource to be requested.
- **bearer_auth** (*bool*) – Whether to use Bearer Authentication or not, defaults to *True*.
- ****req_kwargs** (*dict*) – Keyworded args to be passed down to Requests.

resolve_redirects(*resp*, *req*, *stream=False*, *timeout=None*, *verify=True*, *cert=None*, *proxies=None*, *yield_requests=False*, ***adapter_kwargs*)
Receives a Response. Returns a generator of Responses or Requests.

send(*request*, ***kwargs*)
Send a given PreparedRequest.

Return type requests.Response

Ofly Sessions

class `rauth.`**OflySession**(*app_id*, *app_secret*, *user_id=None*, *service=None*)
A specialized `Session` object, wrapping Ofly logic.

This object is utilized by the *`OflyService`* wrapper but can be used independently of that infrastructure. Essentially this is a loose wrapping around the standard Requests codepath. State may be tracked at this layer, especially if the instance is kept around and tracked via some unique identifier. Things like request cookies will be preserved between requests and in fact all functionality provided by a Requests' `Session` object should be exposed here.

If you were to use this object by itself you could do so by instantiating it like this:

```
session = OflySession('123', '456')
```

You now have a session object which can be used to make requests exactly as you would with a normal Requests `Session` instance. This anticipates that the standard Ofly flow will be modeled outside of the scope of this class. In other words, if the fully qualified flow is useful to you then this object probably need not be used directly, instead consider using *`OflyService`*.

Once the session object is setup, you may start making requests:

```
r = session.get('https://example/com/api/resource',
                params={'format': 'json'})
print r.json()
```

Parameters

- **app_id** (*str*) – The oFlyAppId, i.e. "application ID".
- **app_secret** (*str*) – The oFlyAppSecret, i.e. "shared secret".
- **service** (`rauth.Service`) – A back reference to the service wrapper, defaults to *None*.

app_id = None
Client credentials.

close()
Closes all adapters and as such the session

delete(*url*, ***kwargs*)
Sends a DELETE request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

get(*url*, ***kwargs*)
Sends a GET request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

get_adapter(*url*)
Returns the appropriate connection adapter for the given URL.

Return type requests.adapters.BaseAdapter

get_redirect_target(*resp*)
Receives a Response. Returns a redirect URI or `None`

head(*url*, ***kwargs*)
Sends a HEAD request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

merge_environment_settings(*url*, *proxies*, *stream*, *verify*, *cert*)
Check the environment and merge it with some settings.

Return type dict

mount(*prefix*, *adapter*)
Registers a connection adapter to a prefix.

Adapters are sorted in descending order by prefix length.

options(*url*, ***kwargs*)
Sends a OPTIONS request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

patch(*url*, *data=None*, ***kwargs*)
Sends a PATCH request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

post(*url*, *data=None*, *json=None*, ***kwargs*)
Sends a POST request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- **json** – (optional) json to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

prepare_request(*request*)
Constructs a `PreparedRequest` for transmission and returns it. The `PreparedRequest` has settings merged from the `Request` instance and those of the `Session`.

Parameters **request** – `Request` instance to prepare with this session's settings.

Return type requests.PreparedRequest

put(*url*, *data=None*, ***kwargs*)
Sends a PUT request. Returns `Response` object.

Parameters

- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Return type requests.Response

rebuild_auth(*prepared_request*, *response*)
When being redirected we may want to strip authentication from the request to avoid leaking credentials. This method intelligently removes and reapplies authentication where possible to avoid credential loss.

rebuild_method(*prepared_request*, *response*)
When being redirected we may want to change the method of the request based on certain specs or browser behavior.

rebuild_proxies(*prepared_request*, *proxies*)
This method re-evaluates the proxy configuration by considering the environment variables. If we are redirected to a URL covered by NO_PROXY, we strip the proxy configuration. Otherwise, we set missing proxy keys for this URL (in case they were stripped by a previous redirect).

This method also replaces the Proxy-Authorization header where necessary.

Return type dict

request(*method*, *url*, *user_id=None*, *hash_meth='sha1'*, ***req_kwargs*)
A loose wrapper around Requests' `Session` which injects Ofly parameters.

Parameters

- **method** (*str*) – A string representation of the HTTP method to be used.
- **url** (*str*) – The resource to be requested.
- **hash_meth** (*str*) – The hash method to use for signing, defaults to "sha1".
- **user_id** (*str*) – The oflyUserid, defaults to *None*.
- ****req_kwargs** (*dict*) – Keyworded args to be passed down to Requests.

resolve_redirects(*resp*, *req*, *stream=False*, *timeout=None*, *verify=True*, *cert=None*, *proxies=None*, *yield_requests=False*, ***adapter_kwargs*)
Receives a Response. Returns a generator of Responses or Requests.

send(*request*, ***kwargs*)
Send a given PreparedRequest.

Return type requests.Response

static sign(*url*, *app_id*, *app_secret*, *hash_meth='sha1'*, ***params*)
A signature method which generates the necessary Ofly parameters.

Parameters

- **app_id** (*str*) – The oFlyAppId, i.e. "application ID".
- **app_secret** (*str*) – The oFlyAppSecret, i.e. "shared secret".
- **hash_meth** (*str*) – The hash method to use for signing, defaults to "sha1".
- ****params** (*dict*) – Additional parameters.

user_id = None
oFlyUserid

CHAPTER 4

Upgrade Notes

Notes for upgrading from release to release, if applicable.

Upgrading Rauth

Rauth is continually being improved upon. Sometimes these improvements require breaking changes from release to release. Herein we document these changes and steps you can take to port your code to newer releases.

In order to upgrade you may use:

```
$ pip install -U rauth
```

or:

```
$ easy_install -U rauth
```

Version 0.5.0

This release will bring support for Requests v1.x to rauth. The changes in Requests API are fairly significant and as a direct result the changes to the rauth API in this release are extensive.

First and foremost Requests v1.x largely does away with hooks (and removes the specific hook rauth was previously relying on). As such we have completely moved away from the hook infrastructure and have replaced it with custom Session objects. These objects offer some nice benefits such as keep-alive.

Service wrappers have been restructured to produce instances of their respective Session objects. This is done via the `Session.get_session()` and `Session.get_auth_session()` methods. In particular, *get_auth_session* should be used where possible to retrieve an access token over a *Session* instance. This method returns a session object which is used to make requests. This is in contrast to previous versions of rauth which provided a *request* method on the *Service* wrappers. This method is now gone and all HTTP methods are provided by the *Session* objects instead.

OAuth2Service no longer accepts *consumer_id* and *consumer_secret* in place of *client_id* and *client_secret*. You must update your code if you were using the old names. This is because the OAuth 2 spec defines these names very

clearly. We had previously used the same names as the OAuth1Service wrapper in order to remain consistent between wrappers. However this not inline with the spec and has been deprecated since 0.4.x.

Importantly, service wrappers have done away with almost all *ad hoc* named arguments. This means that grant types, response codes, and other, often required, OAuth parameters are **not** provided by default. These were removed because there were too many special cases and the code became unmanagable. Specifically there are cases where some parameters are required but others where these parameters become optional: we can't resonably handle every case in the library. Instead the consumer should try to manage this themselves by passing in the required parameters explicitly. This is mostly only applicable to OAuth2. That said some of these may be added back in where appropriate. While porting code, be aware that you must be explicit about these parameters.

Additionally there are changes to Requests itself which are mostly beyond the scope of this document. However it is worth noting you can parse a JSON response via *r.json()*. The examples have been updated to demonstrate this.

It may be instructive to reference the examples when updating your applications for use with rauth 0.5.0. There are examples for OAuth 1.0/a and OAuth 2.0 which should be fully functional and which you can run yourself and experiment with.

Rauth Changelog

This provides a list of changes to rauth by release.

Changes in Version 0.7.3

- Moved canonical fork to maxcountryman/rauth
- Fixed Python 3.6 dict bug #1
- Added Python 3.4 support #2

Changes in Version 0.7.2

- Fixed encoding of tilde bug in urlencode #186
- Fixed overriding of Content-Type #166
- Allow optional query parameters #175

Changes in Version 0.7.1

- Fixed Requests token saving in headers #137
- Fixed character encoding; all percent-encoded #165
- Added PLAINTEXT signature support #147

Changes in Version 0.7.0

- Made OAuth1 nonces more secure via SystemRandom
- Exposed authentication responses
- Allowed Requests versions >= 1.2.3
- Fixed OAuth1 unicode encoding issues

Changes in Version 0.6.2

- Made OAuth access token name dynamic

Changes in Version 0.6.1

- Updated PyPI metadata.

Changes in Version 0.6.0

- Added Python 3 support (thanks to everyone who contributed, espcially @sashahart)
- Made service and session objects serializable

Changes in Version 0.5.5

- BUGFIX Fixed upstream CaseInsensitiveDict API changes
- BUGFIX Corrected multiple quoting of oauth_token

Changes in Version 0.5.4

- BUGFIX Corrected adding header to OAuth 2.0 where no access token existed

Changes in Version 0.5.3

- Added an ad hoc check against double signing in OAuth 1.0/a

Changes in Version 0.5.2

- Added ability to pass in key name for get_token methods (excluding raw)
- Added ability to pass in custom decoder for get_token methods (excluding raw)
- Added better error reporting for get_token methods (excluding raw)
- BUGIFX Corrected assignment of custom signature objects in OAuth 1.0 session objects
- Added custom decoder param for request and access token getters
- Updated test runner to not fail when yanc is missing
- Removed bash requirement from test runner
- Updated documentation to include CHANGELOG
- Updated docstring to correct incorrect documentation
- BUGFIX Corrected improper OAuth 1.0/a handling of entity-methods

Changes in Version 0.5.1

- BUGFIX Added CaseInsensitiveDict to ensure headers are properly updated

Changes in Version 0.5.0

- Added CHANGELOG
- Added requirements.txt
- Updated documentation
- Updated README
- All HTTP methods moved to Session objects
- Added get_session method to service wrappers
- Added get_auth_session method to service wrappers
- Added get_raw_access token to OAuth1 and OAuth2Service
- OAuth2 client_id parameter interpolation moved to OAuth2Session
- Default OAuth parameter setting removed from all wrapper methods
- Default redirect_uri removed from all service wrapper helper methods
- Default response_type removed from OAuth2Service.get_authorize_url
- Default oauth_callback removed from OAuth1Service.get_raw_request_token
- parse_utf8_qsl moved out of all get_raw_* Service wrapper methods
- Default remote_user removed from OflyService.get_authorize_url
- Added Ofly MD5 signing option
- Moved Ofly signing logic into staticmethod on OflySession
- raise_for_status removed from all service wrapper helper methods
- Cleaned up OAuth2Service.get_access_token
- Default basic auth removed from OAuth2Service.get_access_token
- Service wrapper API unified
- hook module removed, replaced by session module
- Added unified Requests Session wrappers for each respective service
- Default connection timeouts moved into Session module logic
- All request injection now happens on Session objects
- Removed examples without correct, functioning credentials
- Test suite completely rewritten for more robust request checking

Python Module Index

r

rauth, 7

Index

A

access_token (rauth.OAuth1Session attribute), 14
access_token (rauth.OAuth2Session attribute), 17
access_token_key (rauth.OAuth2Session attribute), 17
access_token_response (rauth.OAuth2Service attribute), 11
access_token_url (rauth.OAuth2Service attribute), 11
app_id (rauth.OflyService attribute), 13
app_id (rauth.OflySession attribute), 20

C

client_id (rauth.OAuth2Service attribute), 11
client_id (rauth.OAuth2Session attribute), 18
close() (rauth.OAuth1Session method), 14
close() (rauth.OAuth2Session method), 18
close() (rauth.OflySession method), 20
consumer_key (rauth.OAuth1Service attribute), 8
consumer_key (rauth.OAuth1Session attribute), 14

D

delete() (rauth.OAuth1Session method), 14
delete() (rauth.OAuth2Session method), 18
delete() (rauth.OflySession method), 21

G

get() (rauth.OAuth1Session method), 15
get() (rauth.OAuth2Session method), 18
get() (rauth.OflySession method), 21
get_access_token() (rauth.OAuth1Service method), 8
get_access_token() (rauth.OAuth2Service method), 11
get_adapter() (rauth.OAuth1Session method), 15
get_adapter() (rauth.OAuth2Session method), 18
get_adapter() (rauth.OflySession method), 21
get_auth_session() (rauth.OAuth1Service method), 9
get_auth_session() (rauth.OAuth2Service method), 12
get_auth_session() (rauth.OflyService method), 13
get_authorize_url() (rauth.OAuth1Service method), 9
get_authorize_url() (rauth.OAuth2Service method), 12
get_authorize_url() (rauth.OflyService method), 13
get_raw_access_token() (rauth.OAuth1Service method), 9
get_raw_access_token() (rauth.OAuth2Service method), 12
get_raw_request_token() (rauth.OAuth1Service method), 10
get_redirect_target() (rauth.OAuth1Session method), 15
get_redirect_target() (rauth.OAuth2Session method), 18
get_redirect_target() (rauth.OflySession method), 21
get_request_token() (rauth.OAuth1Service method), 10
get_session() (rauth.OAuth1Service method), 10
get_session() (rauth.OAuth2Service method), 12
get_session() (rauth.OflyService method), 13

H

head() (rauth.OAuth1Session method), 15
head() (rauth.OAuth2Session method), 18
head() (rauth.OflySession method), 21

M

merge_environment_settings() (rauth.OAuth1Session method), 15
merge_environment_settings() (rauth.OAuth2Session method), 18
merge_environment_settings() (rauth.OflySession method), 21
mount() (rauth.OAuth1Session method), 15
mount() (rauth.OAuth2Session method), 18
mount() (rauth.OflySession method), 21

O

OAuth1Service (class in rauth), 7
OAuth1Session (class in rauth), 14
OAuth2Service (class in rauth), 10
OAuth2Session (class in rauth), 17
OflyService (class in rauth), 12
OflySession (class in rauth), 20
options() (rauth.OAuth1Session method), 15
options() (rauth.OAuth2Session method), 18

options() (rauth.OflySession method), 21

P

patch() (rauth.OAuth1Session method), 15
patch() (rauth.OAuth2Session method), 18
patch() (rauth.OflySession method), 21
post() (rauth.OAuth1Session method), 15
post() (rauth.OAuth2Session method), 19
post() (rauth.OflySession method), 22
prepare_request() (rauth.OAuth1Session method), 16
prepare_request() (rauth.OAuth2Session method), 19
prepare_request() (rauth.OflySession method), 22
put() (rauth.OAuth1Session method), 16
put() (rauth.OAuth2Session method), 19
put() (rauth.OflySession method), 22

R

rauth (module), 7
rebuild_auth() (rauth.OAuth1Session method), 16
rebuild_auth() (rauth.OAuth2Session method), 19
rebuild_auth() (rauth.OflySession method), 22
rebuild_method() (rauth.OAuth1Session method), 16
rebuild_method() (rauth.OAuth2Session method), 19
rebuild_method() (rauth.OflySession method), 22
rebuild_proxies() (rauth.OAuth1Session method), 16
rebuild_proxies() (rauth.OAuth2Session method), 19
rebuild_proxies() (rauth.OflySession method), 22
request() (rauth.OAuth1Session method), 16
request() (rauth.OAuth2Session method), 20
request() (rauth.OflySession method), 22
request_token_response (rauth.OAuth1Service attribute), 10
request_token_url (rauth.OAuth1Service attribute), 10
resolve_redirects() (rauth.OAuth1Session method), 17
resolve_redirects() (rauth.OAuth2Session method), 20
resolve_redirects() (rauth.OflySession method), 23

S

send() (rauth.OAuth1Session method), 17
send() (rauth.OAuth2Session method), 20
send() (rauth.OflySession method), 23
session_obj (rauth.OAuth1Service attribute), 10
session_obj (rauth.OAuth2Service attribute), 12
session_obj (rauth.OflyService attribute), 13
sign() (rauth.OflySession static method), 23
signature (rauth.OAuth1Session attribute), 17
signature_obj (rauth.OAuth1Service attribute), 10

U

user_id (rauth.OflyService attribute), 13
user_id (rauth.OflySession attribute), 23