mygpoclient Documentation

Release 1.7

gPodder Team

Jun 23, 2022

Contents:

1 A gpodder.net client in Python **1**
- 1.1 Installation . . . 1
- 1.2 Tutorial . . . 2
- 1.3 The mygpoclient package . . . 6

2 Indices and tables **19**

Python Module Index **21**

Index **23**

CHAPTER 1

A gpodder.net client in Python

This reference manual describes how to utilize the mygpoclient API Client Library to interface with the gpodder.net web service. The client library provides a well-tested implementation that adheres to the.

This reference manual describes how to utilize the `mygpoclient` API Client Library to interface with the gpodder.net web service. The client library provides a well-tested implementation that adheres to the.

1.1 Installation

This section describes how to obtain the source code and carry out basic tests to make sure your system has been set up correctly and all dependencies are fulfilled.

1.1.1 Getting the code

The code for the client library is hosted at https://github.com/gpodder/mygpoclient/. You can download the latest version of the code by using the command:

```
git clone https://github.com/gpodder/mygpoclient.git
```

or install it with `pip`:

```
pip install mygpoclient
```

1.1.2 Running Unit tests

To make sure that the library is working and all dependencies are installed, please install the dependencies listed in the DEPENDENCIES file. After that, you can easily run the unit tests that come with the library:

```
make test
```

This will run all unit tests and doctests in the library. After the tests have been completed, you should get a summary with the test and code coverage statistics. The output should look like the following example if everything works (please note that the statement count could be different as development of the library continues):

```
Name                    Stmts   Exec  Cover   Missing
-------------------------------------------------------
mygpoclient                 5      5   100%
mygpoclient.api           155    155   100%
mygpoclient.http           52     52   100%
mygpoclient.json           22     22   100%
mygpoclient.locator        52     52   100%
mygpoclient.simple         16     16   100%
mygpoclient.util           20     20   100%
-------------------------------------------------------
TOTAL                     322    322   100%
-------------------------------------------------------
Ran 81 tests in 4.987s
```

1.1.3 Reading the module documentation

You can use the `pydoc` utility to read the documentation for the library. You probably want to use the following commands:

```
pydoc mygpoclient.simple
pydoc mygpoclient.api
```

If you want, you can let Epydoc create the API documentation in the source tree:

```
make docs
```

The resulting documentation will be created in the `docs/` subdirectory.

1.2 Tutorial

This section gives a short how-to on how to use the library to interact with the web service.

1.2.1 The Simple API client (`mygpoclient.simple.SimpleClient`)

The *`SimpleClient`* class is the most basic way of interacting with the gpodder.net web service. You can use it to provide quick integration of an existing application or in cases where your client code has to be stateless. Please use the Advanced API client (which has a superset of *`SimpleClient`*'s features) for more advanced use cases.

Importing the module

Make sure that the *`mygpoclient`* package is in your `sys.path`. You can set the `PYTHONPATH` environment variable to the Git checkout folder or add it to the `sys.path` list in Python. After that, you can import the `simple` module of the `mygpoclient` package:

```
from mygpoclient import simple
```

1.2.2 Creating a `SimpleClient` instance

The client provides access to user-specific data, so you need to have the username and password of the user you want to authenticate as ready. Also, as gpodder.net is open source, and can potentially run on hosts different from gpodder.net, you can optionally provide the hostname.

Let's suppose we want to authenticate as user `john` and the password `secret` to the default web service (that's '"gpodder.net"'). To create a client object, we would use the following code:

```
client = simple.SimpleClient('john', 'secret')
```

If you have the web service running on another host (for example on port `1337` on the local host or `localhost`), you can specify the host and port as third argument to the *`SimpleClient`* constructor (but you still need to provide username and password in this case):

```
client = simple.SimpleClient('john', 'secret', 'localhost:1337')
```

Downloading subscription lists

You can download a list of podcast subscriptions with *SimpleClient.get_subscriptions(device_id)*. The given *device_id* has to exist for the logged-in user. If the device does not exist, a *`mygpoclient.http.NotFound`* exception is raised.

To download the subscription list of the device `legacy`, use:

```
subscriptions = client.get_subscriptions('legacy')
```

The resulting list contains URLs of all the subscribed podcasts for this device:

```
for url in subscriptions:
    print 'Subscribed to:', url
```

Uploading subscription lists

As with the download function, you can also upload subscriptions. For this, you need to use the `SimpleClient.put_subscriptions(device_id, urls)` function. The function returns `True` if the upload operation was successful, or False otherwise. An example usage would be like this:

```
subscriptions = []
subscriptions.append('http://example.org/episodes.rss')
subscriptions.append('http://example.com/feeds/podcast.xml')
client.put_subscriptions('gpodder-example-device', subscriptions)
```

The existing list of subscriptions is always overwritten, and the user's subscription history will mark all podcasts that have been in the list before but have been removed from the uploaded subscription list as unsubscribed.

Putting it all together (complete example)

Now that we have discussed the basics, we can write a simple but feature-complete command-line application for downloading and uploading subscription lists (this code can also be found in the source tree as simple-client:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# gpodder.net API Client
# Copyright (C) 2009-2013 Thomas Perl and the gPodder Team
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.


from __future__ import print_function
import sys
import getpass

import mygpoclient

from mygpoclient import simple


def usage():
    print("""
    Usage: python %s {get|put} {username} {device_id} [host_or_url]
    """ % (sys.argv[0],), file=sys.stderr)
    sys.exit(1)


if __name__ == '__main__':
    # Use the default url if not specified
    if len(sys.argv) == 4:
        sys.argv.append(mygpoclient.ROOT_URL)

    # Check for valid arguments
    if len(sys.argv) != 5:
        usage()

    # Split arguments in local variables
    progname, subcommand, username, device_id, root_url = sys.argv

    # Check for valid subcommand
    if subcommand not in ('get', 'put'):
        usage()

    # Read password from the terminal
    password = getpass.getpass("%s@%s's password: " % (username, root_url))

    # Create the client object with username/password/root_url set
    client = simple.SimpleClient(username, password, root_url)

```

(continues on next page)

(continued from previous page)

```
    if subcommand == 'get':
        # Get the URL list and print it, one per line
        print('\n'.join(client.get_subscriptions(device_id)))
    elif subcommand == 'put':
        # Read the URL list from standard input and upload it
        print('Enter podcast URLs, one per line.', file=sys.stderr)
        urls = sys.stdin.read().splitlines()
        if client.put_subscriptions(device_id, urls):
            print('Upload successful.', file=sys.stderr)
        else:
            print('Could not upload list.', file=sys.stderr)
```

1.2.3 The Advanced API client (`mygpoclient.api.MygPodderClient`)

The *MygPodderClient* class inherits from *SimpleClient*, so you can use both the Simple API and Advanced API functionality with this class.

Working with subscription lists

Given a device ID, you can update the list of subscriptions via the *update_subscriptions()* method. You need to pass a device_id and two lists (one for the URLs that have been added and one for the URLs that have been removed) to update the subscriptions on the server, for example:

```
from mygpoclient import api

client = api.MygPodderClient('myusername', 'S3cr3tPassw0rd')

to_add = ['http://lugradio.org/episodes.rss']
to_remove = ['http://example.org/episodes.rss',
             'http://example.com/feed.xml']
result = client.update_subscriptions('abcdevice', to_add, to_remove)
```

You can use empty lists if you just add or remove items. As you can see in the code example, the function (if successful) returns a UpdateResult object. The UpdateResult object contains a *update_urls* attribute that is a list of *(old_url, new_url)* tuples in case the server has re-written URLs. According to the API documentation, the client application must update the old_url values with new_url values (these have been sanitized). The *since* attribute of the result object can be used for requesting changes since the last upload, like this:

```
updates = client.pull_subscriptions('abcdevice', result.since)
```

The *updates* return value here is a *SubscriptionChanges* object that contains a new `since` attribute (that can be used for subsequent requests) and two lists (`add` and `remove`) of URLs that should be processed by the client by creating and removing subscriptions.

Synchronizing episode actions

''TODO"

Enumerating and modifying devices

The *MygPodderClient* class provides two methods for dealing with devices: *get_devices()* (which returns a list of `PodcastDevice` objects) and *update_device_settings()* for modifying the device-specific settings

on the server side. Here is an example script to use both functions to rename all users's device to "My Device" on the server:

```
client = api.MygPodderClient('john', '53cr3t')

for device in client.get_devices():
    print 'Changing name of', device.device_id
    client.update_device_settings(device.device_id, caption='My Device')
```

1.2.4 The Public API client (`mygpoclient.public.PublicClient`)

This client does not need authentication and can be used to query and retrieve public data from the web service.

Toplist retrieval

You can use the *`get_toplist()`* method on a *`PublicClient`* instance to retrieve a list of podcast toplist entries:

```
from mygpoclient import public

client = public.PublicClient()

toplist = client.get_toplist()
for index, entry in enumerate(toplist):
    print '%4d. %s (%d)' % (index+1, entry.title, entry.subscribers)
```

Searching for podcasts

Searching is done using *`search_podcasts()`* method of *`PublicClient`* like this:

```
from mygpoclient import public

client = public.PublicClient()

search_results = client.search_podcasts('outlaws')
for result in search_results:
    print result.title
    print result.url
    print '-'*50
```

1.3 The mygpoclient package

1.3.1 Submodules

1.3.2 mygpoclient.api module

class mygpoclient.api.**EpisodeAction**(*podcast*, *episode*, *action*, *device=None*, *timestamp=None*, *started=None*, *position=None*, *total=None*)

Bases: `object`

This class encapsulates an episode action

The mandatory attributes are: podcast - The feed URL of the podcast episode - The enclosure URL or GUID of the episode action - One of 'download', 'play', 'delete' or 'new'

The optional attributes are: device - The device_id on which the action has taken place timestamp - When the action took place (in XML time format) started - The start time of a play event in seconds position - The current position of a play event in seconds total - The total time of the episode (for play events)

The attribute "position" is only valid for "play" action types.

VALID_ACTIONS = ('download', 'play', 'delete', 'new', 'flattr')

classmethod from_dictionary(*d*)

to_dictionary()

class mygpoclient.api.**EpisodeActionChanges**(*actions*, *since*)
Bases: object

Container for added episode actions

Attributes: actions - A list of EpisodeAction objects since - A timestamp value for use in future requests

exception mygpoclient.api.**InvalidResponse**
Bases: exceptions.Exception

class mygpoclient.api.**MygPodderClient**(*username*, *password*, *root_url='http://gpodder.net'*, *client_class=<class 'mygpoclient.json.JsonClient'>*)
Bases: *mygpoclient.simple.SimpleClient*

gpodder.net API Client

This is the API client that implements both the Simple and Advanced API of gpodder.net. See the SimpleClient class for a smaller class that only implements the Simple API.

download_episode_actions(**args*, ***kwargs*)
Downloads a list of EpisodeAction objects from the server

Returns a EpisodeActionChanges object with the list of new actions and a "since" timestamp that can be used for future calls to this method when retrieving episodes.

get_devices(**args*, ***kwargs*)
Returns a list of this user's PodcastDevice objects

The resulting list can be used to display a selection list to the user or to determine device IDs to pull the subscription list from.

get_favorite_episodes()
Returns a List of Episode Objects containing the Users favorite Episodes

get_settings(*type*, *scope_param1=None*, *scope_param2=None*)
Returns a Dictionary with the set settings for the type & specified scope

get_subscriptions(**args*, ***kwargs*)
Get a list of subscriptions for a device

Returns a list of URLs (one per subscription) for the given device_id that reflects the current list of subscriptions.

Raises http.NotFound if the device does not exist.

pull_subscriptions(**args*, ***kwargs*)
Downloads subscriptions since the time of the last update

The "since" parameter should be a timestamp that has been retrieved previously by a call to update_subscriptions or pull_subscriptions.

Returns a SubscriptionChanges object with two lists (one for added and one for removed podcast URLs) and a "since" value that can be used for future calls to this method.

put_subscriptions(**args*, ***kwargs*)
Update a device's subscription list

Sets the server-side subscription list for the device "device_id" to be equivalent to the URLs in the list of strings "urls".

The device will be created if it does not yet exist.

Returns True if the update was successful, False otherwise.

set_settings(*type*, *scope_param1*, *scope_param2*, *set={}*, *remove=[]*)
Returns a Dictionary with the set settings for the type & specified scope

update_device_settings(**args*, ***kwargs*)
Update the description of a device on the server

This changes the caption and/or type of a given device on the server. If the device does not exist, it is created with the given settings.

The parameters caption and type are both optional and when set to a value other than None will be used to update the device settings.

Returns True if the request succeeded, False otherwise.

update_subscriptions(**args*, ***kwargs*)
Update the subscription list for a given device.

Returns a UpdateResult object that contains a list of (sanitized) URLs and a "since" value that can be used for future calls to pull_subscriptions.

For every (old_url, new_url) tuple in the updated_urls list of the resulting object, the client should rewrite the URL in its subscription list so that new_url is used instead of old_url.

upload_episode_actions(**args*, ***kwargs*)
Uploads a list of EpisodeAction objects to the server

Returns the timestamp that can be used for retrieving changes.

class mygpoclient.api.**PodcastDevice**(*device_id*, *caption*, *type*, *subscriptions*)
Bases: object

This class encapsulates a podcast device

Attributes: device_id - The ID used to refer to this device caption - A user-defined "name" for this device type - A valid type of podcast device (see VALID_TYPES) subscriptions - The number of podcasts this device is subscribed to

VALID_TYPES = ('desktop', 'laptop', 'mobile', 'server', 'other')

classmethod from_dictionary(*d*)

class mygpoclient.api.**SubscriptionChanges**(*add*, *remove*, *since*)
Bases: object

Container for subscription changes

Attributes: add - A list of URLs that have been added remove - A list of URLs that have been removed since - A timestamp value for use in future requests

class mygpoclient.api.**UpdateResult**(*update_urls*, *since*)
Bases: object

Container for subscription update results

Attributes: update_urls - A list of (old_url, new_url) tuples since - A timestamp value for use in future requests

1.3.3 mygpoclient.feeds module

class mygpoclient.feeds.**FeedServiceResponse**(*feeds*, *last_modified*, *feed_urls*)
Bases: list

Encapsulates the relevant data of a mygpo-feedservice response

get_feed(*url*)
Returns the parsed feed for the given URL

get_feeds()
Returns the parsed feeds in order of the initial request

class mygpoclient.feeds.**FeedserviceClient**(*username=None*, *password=None*, *base_url='http://mygpo-feedservice.appspot.com'*)
Bases: *mygpoclient.json.JsonClient*

A special-cased JsonClient for mygpo-feedservice

build_url(***kwargs*)
Parameter such as strip_html, scale_logo, etc are pased as kwargs

static format_header_date(*datetime_obj*)
Formats the given datetime object for use in HTTP headers

parse_feeds(*feed_urls*, *last_modified=None*, *strip_html=False*, *use_cache=True*, *inline_logo=False*, *scale_logo=None*, *logo_format=None*)
Passes the given feed-urls to mygpo-feedservice to be parsed and returns the response

static parse_header_date(*date_str*)
Parses dates in RFC2822 format to datetime objects

1.3.4 mygpoclient.http module

exception mygpoclient.http.**BadRequest**
Bases: exceptions.Exception

class mygpoclient.http.**HttpClient**(*username=None*, *password=None*)
Bases: object

A comfortable HTTP client

This class hides the gory details of the underlying HTTP protocol from the rest of the code by providing a simple interface for doing requests and handling authentication.

GET(*uri*)
Convenience method for carrying out a GET request

POST(*uri*, *data*)
Convenience method for carrying out a POST request

PUT(*uri*, *data*)
Convenience method for carrying out a PUT request

class mygpoclient.http.**HttpRequest**(*url*, *data=None*, *headers={}*, *origin_req_host=None*, *unverifiable=False*)
Bases: urllib2.Request

Request object with customizable method

The default behaviour of Request is unchanged:

```
>>> request = HttpRequest('http://example.org/')
>>> request.get_method()
'GET'
>>> request = HttpRequest('http://example.org/', data='X')
>>> request.get_method()
'POST'
```

However, it's possible to customize the method name:

```
>>> request = HttpRequest('http://example.org/', data='X')
>>> request.set_method('PUT')
>>> request.get_method()
'PUT'
```

get_method()

set_method(*method*)

exception mygpoclient.http.**NotFound**
Bases: exceptions.Exception

class mygpoclient.http.**SimpleHttpPasswordManager**(*username*, *password*)
Bases: urllib2.HTTPPasswordMgr

Simplified password manager for urllib2

This class always provides the username/password combination that is passed to it as constructor argument, independent of the realm or authuri that is used.

MAX_RETRIES = 3

find_user_password(*realm*, *authuri*)

exception mygpoclient.http.**Unauthorized**
Bases: exceptions.Exception

exception mygpoclient.http.**UnknownResponse**
Bases: exceptions.Exception

1.3.5 mygpoclient.json module

class mygpoclient.json.**JsonClient**(*username=None*, *password=None*)
Bases: *mygpoclient.http.HttpClient*

A HttpClient with built-in JSON support

This client will automatically marshal and unmarshal data for JSON-related web services so that code using this class will not need to care about (de-)serialization of data structures.

static decode(*data*)
Decodes a response string to a Python object

```
>>> JsonClient.decode(b'')
>>> JsonClient.decode(b'[1,2,3]')
[1, 2, 3]
>>> JsonClient.decode(b'42')
42
```

static encode(*data*)
Encodes a object into its JSON string repesentation

```
>>> JsonClient.encode(None) is None
True
>>> JsonClient.encode([1,2,3]) == b'[1, 2, 3]'
True
>>> JsonClient.encode(42) == b'42'
True
```

exception mygpoclient.json.**JsonException**
Bases: exceptions.Exception

1.3.6 mygpoclient.locator module

class mygpoclient.locator.**Locator**(*username*, *root_url='http://gpodder.net'*, *version=2*)
Bases: object

URI Locator for API endpoints

This helper class abstracts the URIs for the gpodder.net webservice and provides a nice facility for generating API URIs and checking parameters.

SETTINGS_TYPES = ('account', 'device', 'podcast', 'episode')

SIMPLE_FORMATS = ('opml', 'json', 'txt')

add_remove_subscriptions_uri(*device_id*)
Get the Advanced API URI for uploading list diffs

```
>>> locator = Locator('bill')
>>> locator.add_remove_subscriptions_uri('n810')
'http://gpodder.net/api/2/subscriptions/bill/n810.json'
```

device_list_uri()
Get the Advanced API URI for retrieving the device list

```
>>> locator = Locator('jeff')
>>> locator.device_list_uri()
'http://gpodder.net/api/2/devices/jeff.json'
```

device_settings_uri(*device_id*)
Get the Advanced API URI for setting per-device settings uploads

```
>>> locator = Locator('mike')
>>> locator.device_settings_uri('ipod')
'http://gpodder.net/api/2/devices/mike/ipod.json'
```

download_episode_actions_uri(*since=None*, *podcast=None*, *device_id=None*)
Get the Advanced API URI for downloading episode actions

The parameter "since" is optional and should be a numeric value (otherwise a ValueError is raised).

Both "podcast" and "device_id" are optional and exclusive:

"podcast" should be a podcast URL "device_id" should be a device ID

```
>>> locator = Locator('steve')
>>> locator.download_episode_actions_uri()
'http://gpodder.net/api/2/episodes/steve.json'
>>> locator.download_episode_actions_uri(since=1337)
'http://gpodder.net/api/2/episodes/steve.json?since=1337'
>>> locator.download_episode_actions_uri(podcast='http://example.org/episodes.
↪rss')
'http://gpodder.net/api/2/episodes/steve.json?podcast=http%3A//example.org/
↪episodes.rss'
>>> locator.download_episode_actions_uri(since=2000, podcast='http://example.
↪com/')
'http://gpodder.net/api/2/episodes/steve.json?since=2000&podcast=http%3A//
↪example.com/'
>>> locator.download_episode_actions_uri(device_id='ipod')
'http://gpodder.net/api/2/episodes/steve.json?device=ipod'
>>> locator.download_episode_actions_uri(since=54321, device_id='ipod')
'http://gpodder.net/api/2/episodes/steve.json?since=54321&device=ipod'
```

episode_data_uri (*podcast_url*, *episode_url*)
Get the Advanced API URI for retrieving Episode Data

```
>>> locator = Locator(None)
>>> locator.episode_data_uri('http://podcast.com','http://podcast.com/foo')
'http://gpodder.net/api/2/data/episode.json?podcast=http%3A//podcast.com&
↪url=http%3A//podcast.com/foo'
```

favorite_episodes_uri ()
Get the Advanced API URI for listing favorite episodes

```
>>> locator = Locator('mike')
>>> locator.favorite_episodes_uri()
'http://gpodder.net/api/2/favorites/mike.json'
```

podcast_data_uri (*podcast_url*)
Get the Advanced API URI for retrieving Podcast Data

```
>>> locator = Locator(None)
>>> locator.podcast_data_uri('http://podcast.com')
'http://gpodder.net/api/2/data/podcast.json?url=http%3A//podcast.com'
```

podcasts_of_a_tag_uri (*tag*, *count=50*)
Get the Advanced API URI for retrieving the top Podcasts of a Tag

```
>>> locator = Locator(None)
>>> locator.podcasts_of_a_tag_uri('linux')
'http://gpodder.net/api/2/tag/linux/50.json'
>>> locator.podcasts_of_a_tag_uri('linux',70)
'http://gpodder.net/api/2/tag/linux/70.json'
```

root_uri ()
Get the server's root URI.

```
>>> locator = Locator(None)
>>> locator.root_uri()
'http://gpodder.net'
```

search_uri (*query*, *format='opml'*)
Get the Simple API URI for podcast search

```
>>> locator = Locator(None)
>>> locator.search_uri('outlaws')
'http://gpodder.net/search.opml?q=outlaws'
>>> locator.search_uri(':something?', 'txt')
'http://gpodder.net/search.txt?q=%3Asomething%3F'
>>> locator.search_uri('software engineering', 'json')
'http://gpodder.net/search.json?q=software+engineering'
```

settings_uri (*type*, *scope_param1*, *scope_param2*)
Get the Advanced API URI for retrieving or saving Settings

Depending on the Type of setting scope_param2 or scope_param1 and scope_param2 are not necessary.

```
>>> locator = Locator('joe')
>>> locator.settings_uri('account',None,None)
'http://gpodder.net/api/2/settings/joe/account.json'
>>> locator.settings_uri('device','foodevice',None)
'http://gpodder.net/api/2/settings/joe/device.json?device=foodevice'
>>> locator.settings_uri('podcast','http://podcast.com',None)
'http://gpodder.net/api/2/settings/joe/podcast.json?podcast=http%3A//podcast.
↪com'
>>> locator.settings_uri('episode','http://podcast.com','http://podcast.com/
↪foo')
'http://gpodder.net/api/2/settings/joe/episode.json?podcast=http%3A//podcast.
↪com&episode=http%3A//podcast.com/foo'
```

subscription_updates_uri (*device_id*, *since=None*)
Get the Advanced API URI for downloading list diffs

The parameter "since" is optional and should be a numeric value (otherwise a ValueError is raised).

```
>>> locator = Locator('jen')
>>> locator.subscription_updates_uri('n900')
'http://gpodder.net/api/2/subscriptions/jen/n900.json'
>>> locator.subscription_updates_uri('n900', 1234)
'http://gpodder.net/api/2/subscriptions/jen/n900.json?since=1234'
```

subscriptions_uri (*device_id=None*, *format='opml'*)
Get the Simple API URI for a subscription list

```
>>> locator = Locator('john')
>>> locator.subscriptions_uri('n800')
'http://gpodder.net/subscriptions/john/n800.opml'
>>> locator.subscriptions_uri('ipod', 'txt')
'http://gpodder.net/subscriptions/john/ipod.txt'
```

suggestions_uri (*count=10*, *format='opml'*)
Get the Simple API URI for user suggestions

```
>>> locator = Locator('john')
>>> locator.suggestions_uri()
'http://gpodder.net/suggestions/10.opml'
>>> locator.suggestions_uri(50)
'http://gpodder.net/suggestions/50.opml'
>>> locator.suggestions_uri(70, 'json')
'http://gpodder.net/suggestions/70.json'
```

toplist_uri(*count=50*, *format='opml'*)
Get the Simple API URI for the toplist

```
>>> locator = Locator(None)
>>> locator.toplist_uri()
'http://gpodder.net/toplist/50.opml'
>>> locator.toplist_uri(70)
'http://gpodder.net/toplist/70.opml'
>>> locator.toplist_uri(10, 'json')
'http://gpodder.net/toplist/10.json'
```

toptags_uri(*count=50*)
Get the Advanced API URI for retrieving the top Tags

```
>>> locator = Locator(None)
>>> locator.toptags_uri()
'http://gpodder.net/api/2/tags/50.json'
>>> locator.toptags_uri(70)
'http://gpodder.net/api/2/tags/70.json'
```

upload_episode_actions_uri()
Get the Advanced API URI for uploading episode actions

```
>>> locator = Locator('thp')
>>> locator.upload_episode_actions_uri()
'http://gpodder.net/api/2/episodes/thp.json'
```

1.3.7 mygpoclient.public module

class mygpoclient.public.**Episode**(*title*, *url*, *podcast_title*, *podcast_url*, *description*, *website*, *released*, *mygpo_link*)

Bases: object

Container Class for Episodes

Attributes: title - url - podcast_title - podcast_url - description - website - released - mygpo_link -

REQUIRED_KEYS = ('title', 'url', 'podcast_title', 'podcast_url', 'description', 'websi

classmethod from_dict(*d*)

class mygpoclient.public.**PublicClient**(*root_url='http://gpodder.net'*, *client_class=<class 'mygpoclient.json.JsonClient'>*)

Bases: object

Client for the gpodder.net "anonymous" API

This is the API client implementation that provides a pythonic interface to the parts of the gpodder.net Simple API that don't need user authentication.

FORMAT = 'json'

get_episode_data(*podcast_uri*, *episode_uri*)
Get Metadata for the specified Episode

Returns a Episode object.

The parameter "podcast_uri" specifies the URL of the Podcast, which this Episode belongs to

The parameter "episode_uri" specifies the URL of the Episode

get_podcast_data(*podcast_uri*)
Get Metadata for the specified Podcast

Returns a simple.Podcast object.

The parameter "podcast_uri" specifies the URL of the Podcast.

get_podcasts_of_a_tag(*tag*, *count=50*)
Get a list of most-subscribed podcasts of a Tag

Returns a list of simple.Podcast objects.

The parameter "tag" specifies the tag as a String

The parameter "count" is optional and describes the amount of podcasts that are returned. The default value is 50, the minimum value is 1 and the maximum value is 100.

get_toplist(*count=50*)
Get a list of most-subscribed podcasts

Returns a list of simple.Podcast objects.

The parameter "count" is optional and describes the amount of podcasts that are returned. The default value is 50, the minimum value is 1 and the maximum value is 100.

get_toptags(*count=50*)
Get a list of most-used tags

Returns a list of Tag objects.

The parameter "count" is optional and describes the amount of podcasts that are returned. The default value is 50, the minimum value is 1 and the maximum value is 100.

search_podcasts(*query*)
Search for podcasts on the webservice

Returns a list of simple.Podcast objects.

The parameter "query" specifies the search query as a string.

class mygpoclient.public.**Tag**(*tag*, *usage*)
Bases: object

Container class for a tag in the top tag list

Attributes: tag - The name of the tag usage - Usage of the tag

REQUIRED_KEYS = ('tag', 'usage')

classmethod from_dict(*d*)

1.3.8 mygpoclient.simple module

exception mygpoclient.simple.**MissingCredentials**
Bases: exceptions.Exception

Raised when instantiating a SimpleClient without credentials

class mygpoclient.simple.**Podcast**(*url*, *title*, *description*, *website*, *subscribers*, *subscribers_last_week*, *mygpo_link*, *logo_url*)
Bases: object

Container class for a podcast

Encapsulates the metadata for a podcast.

Attributes: url - The URL of the podcast feed title - The title of the podcast description - The description of the podcast

REQUIRED_FIELDS = ('url', 'title', 'description', 'website', 'subscribers', 'subscribe

classmethod from_dict(*d*)

class mygpoclient.simple.**SimpleClient**(*username*, *password*, *root_url='http://gpodder.net'*, *client_class=<class 'mygpoclient.json.JsonClient'>*)

Bases: object

Client for the gpodder.net Simple API

This is the API client implementation that provides a pythonic interface to the gpodder.net Simple API.

FORMAT = 'json'

get_subscriptions(**args*, ***kwargs*)

Get a list of subscriptions for a device

Returns a list of URLs (one per subscription) for the given device_id that reflects the current list of subscriptions.

Raises http.NotFound if the device does not exist.

get_suggestions(**args*, ***kwargs*)

Get podcast suggestions for the user

Returns a list of Podcast objects that are to be suggested to the user.

The parameter count is optional and if specified has to be a value between 1 and 100 (with 10 being the default), and determines how much search results are returned (at maximum).

locator

read-only access to the locator

put_subscriptions(**args*, ***kwargs*)

Update a device's subscription list

Sets the server-side subscription list for the device "device_id" to be equivalent to the URLs in the list of strings "urls".

The device will be created if it does not yet exist.

Returns True if the update was successful, False otherwise.

mygpoclient.simple.**needs_credentials**(*f*)

apply to all methods that initiate requests that require credentials

1.3.9 mygpoclient.testing module

class mygpoclient.testing.**FakeJsonClient**

Bases: object

Fake implementation of a JsonClient used for testing

Set the response using response_value and check the list of requests this object got using the requests list.

GET(*uri*)

POST(*uri*, *data*)

PUT(*uri*, *data*)

1.3.10 mygpoclient.util module

mygpoclient.util.**datetime_to_iso8601**(*dt*)
Convert a datetime to a ISO8601-formatted string

```
>>> datetime_to_iso8601(datetime.datetime(2009, 12, 29, 19, 25, 33))
'2009-12-29T19:25:33'
```

mygpoclient.util.**iso8601_to_datetime**(*s*)
Convert a ISO8601-formatted string to datetime

```
>>> iso8601_to_datetime('2009-12-29T19:25:33')
datetime.datetime(2009, 12, 29, 19, 25, 33)
>>> iso8601_to_datetime('2009-12-29T19:25:33.1')
datetime.datetime(2009, 12, 29, 19, 25, 33, 100000)
>>> iso8601_to_datetime('2009-12-29T19:25:33Z')
datetime.datetime(2009, 12, 29, 19, 25, 33)
>>> iso8601_to_datetime('xXxXxXxXxxxxXxxxXxx')
>>>
```

mygpoclient.util.**join**(**args*)
Join separate URL parts to a full URL

mygpoclient.util.**position_to_seconds**(*s*)
Convert a position string to its amount of seconds

```
>>> position_to_seconds('00:00:01')
1
>>> position_to_seconds('00:01:00')
60
>>> position_to_seconds('01:00:00')
3600
>>> position_to_seconds('02:59:59')
10799
>>> position_to_seconds('100:00:00')
360000
```

mygpoclient.util.**seconds_to_position**(*seconds*)
Convert the amount of seconds to a position string

```
>>> seconds_to_position(1)
'00:00:01'
>>> seconds_to_position(60)
'00:01:00'
>>> seconds_to_position(60*60)
'01:00:00'
>>> seconds_to_position(59 + 60*59 + 60*60*2)
'02:59:59'
>>> seconds_to_position(60*60*100)
'100:00:00'
```

1.3.11 Module contents

gpodder.net API Client Library

This is mygpoclient, the gpodder.net API Client Library for Python.

mygpoclient.**require_version**(*minimum_required*)

Require a minimum version of the library

Returns True if the minimum library version constraint is satisfied, False otherwise. Use this to check for newer API methods and changes in the public API as described in NEWS.

```
>>> require_version('1.0')
True
>>> require_version('1.2')
True
>>> require_version(__version__)
True
>>> require_version('99.99')
False
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mygpoclient, 17
mygpoclient.api, 6
mygpoclient.feeds, 9
mygpoclient.http, 9
mygpoclient.json, 10
mygpoclient.locator, 11
mygpoclient.public, 14
mygpoclient.simple, 15
mygpoclient.testing, 16
mygpoclient.util, 17

Index

A

add_remove_subscriptions_uri() (*mygpoclient.locator.Locator method*), 11

B

BadRequest, 9
build_url() (*mygpoclient.feeds.FeedserviceClient method*), 9

D

datetime_to_iso8601() (*in module mygpoclient.util*), 17
decode() (*mygpoclient.json.JsonClient static method*), 10
device_list_uri() (*mygpoclient.locator.Locator method*), 11
device_settings_uri() (*mygpoclient.locator.Locator method*), 11
download_episode_actions() (*mygpoclient.api.MygPodderClient method*), 7
download_episode_actions_uri() (*mygpoclient.locator.Locator method*), 11

E

encode() (*mygpoclient.json.JsonClient static method*), 10
Episode (*class in mygpoclient.public*), 14
episode_data_uri() (*mygpoclient.locator.Locator method*), 12
EpisodeAction (*class in mygpoclient.api*), 6
EpisodeActionChanges (*class in mygpoclient.api*), 7

F

FakeJsonClient (*class in mygpoclient.testing*), 16
favorite_episodes_uri() (*mygpoclient.locator.Locator method*), 12
FeedserviceClient (*class in mygpoclient.feeds*), 9
FeedServiceResponse (*class in mygpoclient.feeds*), 9
find_user_password() (*mygpoclient.http.SimpleHttpPasswordManager method*), 10
FORMAT (*mygpoclient.public.PublicClient attribute*), 14
FORMAT (*mygpoclient.simple.SimpleClient attribute*), 16
format_header_date() (*mygpoclient.feeds.FeedserviceClient static method*), 9
from_dict() (*mygpoclient.public.Episode class method*), 14
from_dict() (*mygpoclient.public.Tag class method*), 15
from_dict() (*mygpoclient.simple.Podcast class method*), 16
from_dictionary() (*mygpoclient.api.EpisodeAction class method*), 7
from_dictionary() (*mygpoclient.api.PodcastDevice class method*), 8

G

GET() (*mygpoclient.http.HttpClient method*), 9
GET() (*mygpoclient.testing.FakeJsonClient method*), 16
get_devices() (*mygpoclient.api.MygPodderClient method*), 7
get_episode_data() (*mygpoclient.public.PublicClient method*), 14
get_favorite_episodes() (*mygpoclient.api.MygPodderClient method*), 7
get_feed() (*mygpoclient.feeds.FeedServiceResponse method*), 9
get_feeds() (*mygpoclient.feeds.FeedServiceResponse method*), 9
get_method() (*mygpoclient.http.HttpRequest method*), 10

get_podcast_data() (mygpoclient.public.PublicClient method), 14
get_podcasts_of_a_tag() (mygpoclient.public.PublicClient method), 15
get_settings() (mygpoclient.api.MygPodderClient method), 7
get_subscriptions() (mygpoclient.api.MygPodderClient method), 7
get_subscriptions() (mygpoclient.simple.SimpleClient method), 16
get_suggestions() (mygpoclient.simple.SimpleClient method), 16
get_toplist() (mygpoclient.public.PublicClient method), 15
get_toptags() (mygpoclient.public.PublicClient method), 15

H

HttpClient (class in mygpoclient.http), 9
HttpRequest (class in mygpoclient.http), 9

I

InvalidResponse, 7
iso8601_to_datetime() (in module mygpoclient.util), 17

J

join() (in module mygpoclient.util), 17
JsonClient (class in mygpoclient.json), 10
JsonException, 11

L

Locator (class in mygpoclient.locator), 11
locator (mygpoclient.simple.SimpleClient attribute), 16

M

MAX_RETRIES (mygpoclient.http.SimpleHttpPasswordManager attribute), 10
MissingCredentials, 15
mygpoclient (module), 17
mygpoclient.api (module), 6
mygpoclient.feeds (module), 9
mygpoclient.http (module), 9
mygpoclient.json (module), 10
mygpoclient.locator (module), 11
mygpoclient.public (module), 14
mygpoclient.simple (module), 15
mygpoclient.testing (module), 16
mygpoclient.util (module), 17
MygPodderClient (class in mygpoclient.api), 7

N

needs_credentials() (in module mygpoclient.simple), 16
NotFound, 10

P

parse_feeds() (mygpoclient.feeds.FeedserviceClient method), 9
parse_header_date() (mygpoclient.feeds.FeedserviceClient static method), 9
Podcast (class in mygpoclient.simple), 15
podcast_data_uri() (mygpoclient.locator.Locator method), 12
PodcastDevice (class in mygpoclient.api), 8
podcasts_of_a_tag_uri() (mygpoclient.locator.Locator method), 12
position_to_seconds() (in module mygpoclient.util), 17
POST() (mygpoclient.http.HttpClient method), 9
POST() (mygpoclient.testing.FakeJsonClient method), 16
PublicClient (class in mygpoclient.public), 14
pull_subscriptions() (mygpoclient.api.MygPodderClient method), 7
PUT() (mygpoclient.http.HttpClient method), 9
PUT() (mygpoclient.testing.FakeJsonClient method), 16
put_subscriptions() (mygpoclient.api.MygPodderClient method), 8
put_subscriptions() (mygpoclient.simple.SimpleClient method), 16

R

require_version() (in module mygpoclient), 17
REQUIRED_FIELDS (mygpoclient.simple.Podcast attribute), 16
REQUIRED_KEYS (mygpoclient.public.Episode attribute), 14
REQUIRED_KEYS (mygpoclient.public.Tag attribute), 15
root_uri() (mygpoclient.locator.Locator method), 12

S

search_podcasts() (mygpoclient.public.PublicClient method), 15
search_uri() (mygpoclient.locator.Locator method), 12
seconds_to_position() (in module mygpoclient.util), 17
set_method() (mygpoclient.http.HttpRequest method), 10

set_settings() (*mygpoclient.api.MygPodderClient method*), 8
SETTINGS_TYPES (*mygpoclient.locator.Locator attribute*), 11
settings_uri() (*mygpoclient.locator.Locator method*), 13
SIMPLE_FORMATS (*mygpoclient.locator.Locator attribute*), 11
SimpleClient (*class in mygpoclient.simple*), 16
SimpleHttpPasswordManager (*class in mygpoclient.http*), 10
subscription_updates_uri() (*mygpoclient.locator.Locator method*), 13
SubscriptionChanges (*class in mygpoclient.api*), 8
subscriptions_uri() (*mygpoclient.locator.Locator method*), 13
suggestions_uri() (*mygpoclient.locator.Locator method*), 13

T

Tag (*class in mygpoclient.public*), 15
to_dictionary() (*mygpoclient.api.EpisodeAction method*), 7
toplist_uri() (*mygpoclient.locator.Locator method*), 13
toptags_uri() (*mygpoclient.locator.Locator method*), 14

U

Unauthorized, 10
UnknownResponse, 10
update_device_settings() (*mygpoclient.api.MygPodderClient method*), 8
update_subscriptions() (*mygpoclient.api.MygPodderClient method*), 8
UpdateResult (*class in mygpoclient.api*), 8
upload_episode_actions() (*mygpoclient.api.MygPodderClient method*), 8
upload_episode_actions_uri() (*mygpoclient.locator.Locator method*), 14

V

VALID_ACTIONS (*mygpoclient.api.EpisodeAction attribute*), 7
VALID_TYPES (*mygpoclient.api.PodcastDevice attribute*), 8