pyspreedly Documentation

Release 2.0

James Rivett-Carnac

Jul 13, 2017

Contents

1 API **3**

1.1 `api` API 3

2 Objectify **7**

2.1 `objectify` Objectify 7

3 Indices and tables **9**

Python Module Index **11**

Contents:

CHAPTER 1

API

api API

class `pyspreedly.api.`**`Client`**(*token*, *site_name*)

class **`Client`**(*token*, *site_name*)

Create an object to manage queries for a Client on a given site.

Parameters

- **`token`** – API access token for authorization.
- **`site_name`** – the site_name registered with spreedly.

`Client.`**`add_fee`**(*subscriber_id*, *name*, *description*, *group*, *amount*)

`add_fee`(*subscriber_id*, *name*, *description*, *group*, *amount*)

Add a fee to a user with subscriber_id :param subscriber_id: the id of the subscriber :param name: the name of the fee (eg - Excess Bandwidth Charge) :param description: a description of the charge :param group: a group to add this charge too :param amount: the amount the charge is for :returns: the response object

`Client.`**`allow_free_trial`**(*subscriber_id*)

`allow_free_trial`(*subscriber_id*)

programatically allow for a new free trial :param subscriber_id: the id of the subscriber :returns: subscriber data as dictionary if all good, :raises: HTTPError if not so good (non-200)

`Client.`**`change_plan`**(*subscriber_id*, *plan_id*)

`change_plan`(*subscriber_id*, *plan_id*)

Change a subscription to a new plan, needs the user to be activated

subscribe a user to a free trial plan. :param subscriber_id: ID of the subscriber :parma plan_id: subscription plan ID to change to :returns: status code :raises: HTTPError if response status not 200

`Client.`**`cleanup`**`()`

`cleanup``()`

Removes ALL subscribers. NEVER USE IN PRODUCTION! (should only Remove test users...) :returns: status code

`Client.`**`complimentary_time_extensions`**`(`*subscriber_id*, *duration*, *duration_units*`)`

`complimentary_time_extension``(`*subscriber_id*, *duration*, *duration_units*`)`

corrisponds to adding complimentary time extension to a subscriber

`Client.`**`create_complimentary_subscription`**`(`*subscriber_id*, *duration*, *duration_units*, *feature_level*, *start_time=None*, *amount=None*`)`

`create_complimentary_subscription``(`*subscriber_id*, *duration*, *duration_units*, *feature_level*[, *start_time=None*, *amount=None*]`)`

corrisponds to adding corrisponding subscription to a subscriber :param subscriber_id: Subscriber ID :param duration: Duration (unitless) :param duration_units: Unit for above (days, weeks, months i think) :param feature_level string: what feature level this is at :param start_time: If assgining a value for pro-rating purpose, you need this start datetime :type start_time: datetime.datetime or None :param amount: How much this comp is worth :type amount: float or None

`Client.`**`create_subscriber`**`(`*customer_id*, *screen_name*`)`

Creates a subscription :param customer_id: Customer ID :param screen_name: Customer's screen name :returns: Data for created customer :raises: HTTPError if response code isn't 201

`Client.`**`delete_subscriber`**`(`*id*`)`

`delete_subscriber``(`*id*`)`

delete a test subscriber :param id: user id :returns: status code

`Client.`**`get_info`**`(`*subscriber_id*`)`

`get_info``(`*subscriber_id*`)`

Parameters **`subscriber_id`** – Id of subscriber to fetch

Returns Data as dictionary

Raises HTTPError if not 200

`Client.`**`get_or_create_subscriber`**`(`*subscriber_id*, *screen_name*`)`

`get_or_create_subscriber``(`*subscriber_id*, *screen_name*`)`

Tries to get info for a subscriber, else creates a new subscriber

`Client.`**`get_plans`**`()`

get subscription plans for the configured site :returns: data as dict :raises: `HTTPError` if response is not 200

`Client.`**`get_signup_url`**`(`*subscriber_id*, *plan_id*, *screen_name*, *token=None*`)`

`get_signup_url``(`*subscriber_id*, *plan_id*, *screen_name*, *token=None*`)`

Subscribe a user to the site plan on a free trial

subscribe a user to a plan, either trial or not :param subscriber_id: ID of the subscriber :param plan_id: subscription plan ID :param screen_name: user screen name :param token: customer token or None - if passed use the token version

of the url

Returns url for subscription

`Client.`**`query`**`(`*url*, *data=None*, *action='get'*`)`

`query``(`*url*[, *data=None*, *put='get'*]`)`

which has the problem that it doesn't check if there is data for PUT, and is hard to read.

status_codes are not checked here, and should be handled by the caller.

Delete is only supported on test users

Parameters

- **`url`** – the api url you wish to reach (not incuding site/version)
- **`data`** (*`UTF-8 encoded XML or None`*) – the data to send in the request. Default to *None*
- **`action`** – one of 'get', 'post', 'put' and 'delete'. Case insensitive, Default 'get'

Returns response object

Return type `requests` response object

`Client.`**`set_info`**`(`*subscriber_id*, ***kw*`)`

this corrisponds to the update-subscriber action. passed kw args are placed into the xml data (not sure how the -/_ are dealt with though)

There is a design flaw atm where sclient.set_info(sclient.get_info(123)) will not work at all as the keys are all different

`Client.`**`subscribe`**`(`*subscriber_id*, *plan_id=None*`)`

`subscribe``(`*subscriber_id*, *plan_id*`)`

Subscribe a user to the site plan on a free trial

subscribe a user to a free trial plan. :param subscriber_id: ID of the subscriber :parma plan_id: subscription plan ID :returns: dictionary with xml data if all is good :raises: HTTPError if response status not 200

CHAPTER 2

Objectify

`objectify` Objectify

`pyspreedly.objectify.`**`objectify_spreedly`**(*xml*)

Does some high level stuff to the XML tree, and then passes it off to *`parse_element()`* to get the data back as a dictionary. Truth be told it is not really objectifying spreedly, but turning it into a dictionary.

Parameters **`xml`** – xml string or file object. If it is a string, it is turned into `StringIO`.

`pyspreedly.objectify.`**`parse_element`**(*element*)

Recursivly parses an element of the xml node depth first. Turns all xml tags to underscore instead of dashes. Handles all types in *_types* (string, integer datetime, decimal, boolean, array). Every other type is treated as a string. There are some damn odd types in the data passed. Warning - this doesn't check that the data is what it should be, or that stuff is not being added.

Parameters **`element`** – `ElementTree` element.

Returns dictionary of the data (unordered but with correct heirarchy).

Raises `MaximumRecursionDepthExceeded` if you do pass some crazy huge and deap XML tree

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

pyspreedly.api, 3
pyspreedly.objectify, 7

Index

A

add_fee() (pyspreedly.api.Client method), 3
allow_free_trial() (pyspreedly.api.Client method), 3

C

change_plan() (pyspreedly.api.Client method), 3
cleanup() (pyspreedly.api.Client method), 4
Client (class in pyspreedly.api), 3
Client.Client (class in pyspreedly.api), 3
complimentary_time_extension() (pyspreedly.api.Client method), 4
complimentary_time_extensions() (pyspreedly.api.Client method), 4
create_complimentary_subscription() (pyspreedly.api.Client method), 4
create_subscriber() (pyspreedly.api.Client method), 4

D

delete_subscriber() (pyspreedly.api.Client method), 4

G

get_info() (pyspreedly.api.Client method), 4
get_or_create_subscriber() (pyspreedly.api.Client method), 4
get_plans() (pyspreedly.api.Client method), 4
get_signup_url() (pyspreedly.api.Client method), 5

O

objectify_spreedly() (in module pyspreedly.objectify), 7

P

parse_element() (in module pyspreedly.objectify), 7
pyspreedly.api (module), 3
pyspreedly.objectify (module), 7

Q

query() (pyspreedly.api.Client method), 5

S

set_info() (pyspreedly.api.Client method), 5
subscribe() (pyspreedly.api.Client method), 5