dynadotpy Documentation

Release 0.1.0

Chris Jones and Kenneth Love (brack3t)

October 16, 2014

Contents

You can view the code of our project or fork it on Github. Dynadotpy implements all current features of the version 2 API.

CHAPTER 1

Installation

Install from PyPI

```
pip install dynadotpy
```

CHAPTER 2

Overview

Dynadot's API returns all responses in plain text. Basically the CSV format. This wrapper will give you pretty Python objects instead. If there is interest in the project I may add JSON responses as well.

CHAPTER 3

Command Failure

If there is an error with the API, you should always get back a dict with a single key and value.

```
{'error': 'More information about the result. Only used when result is error.'}
```

CHAPTER 4

Search Command

The search command allows you to search for available domain names through Dynadot. You are limited to 100 domain names per call.

Search command args:

- domains - Python list of domains you want to search for.

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.search(domains=["example.com", "example2.com"])
```

A successful response will return a list of dicts. Each dict within the list will contain:

- *domain_param* - The domain parameter. For example: domain0
- *domain* - The domain name searched for.
- *language* - The language if any. For example: spa
- *result* - The result of the search.
- *more_info* - More information about the result. Only used when result is 'error'.

Search command possible results:

```
from dynadotpy.client import SEARCH_RESPONSES

SEARCH_RESPONSES = {
    "yes": "The domain is available.",
    "no": "The domain is not available.",
    "offline": "The central registry for this domain is currently offline.",
    "system_busy": "All connections are busy.",
    "over_quota": "Over quota.",
    "error": "There was a syntax or registry error processing this domain."
}
```

CHAPTER 5

Register Command

Calling the register command will create and process a registration order. You must have enough account balance to pay for the order. You can only register a single domain at a time.

Register command args:

- domain - String of the domain you want to register.
- duration - String/Int of the duration in years you wish to register the domain.

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.register(domain="example.com", duration=1)
```

Response dict will contain:

- *domain* - The domain name you want to register; only one domain can be registered per request.
- *language* - The language tag for the requested domain; only needed if the domain is an IDN.
- *expiration_date* - Expiration date in unix time (milliseconds since midnight UTC of January 1, 1970). Only used when result is 'success'.

Register command possible results

```
from dynadot.client import REGISTER_RESPONSES

REGISTER_RESPONSES = {
    "success": "The domain was successfully registered",
    "not_available": "The domain is not available",
    "insufficient_funds": "Not enough account balance to process this registration",
    "offline": "The central registry for this domain is currently offline",
    "system_busy": "All connections are busy",
    "over_quota": "See quota details below",
    "error": "There was a syntax or registry error processing this request"
}
```

CHAPTER 6

Delete Command

Calling the delete command will delete a domain that is still in the grace period. Your account balance will be credited back the registration fee. Domains with privacy, that have been renewed, or that have been moved to a new account cannot be deleted through the API.

Delete command args:

- domain - String of the domain you wish to delete.

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.delete(domain="example.com", duration=1)
```

Response dict will contain:

- *result* - The result of the delete request.
- *more_info* - More information about the result. Only used when result is 'error'.

Delete command possible results:

```
from dynadotpy.client import DELETE_RESPONSES

DELETE_RESPONSES = {
    "success": "The domain was successfully deleted",
    "grace_expired": "The grace period has already expired",
    "too_soon": "Cannot delete a domain the first hour after registration",
    "offline": "The central registry for this domain is currently offline",
    "error": "There was a syntax or registry error processing this request"
}
```

CHAPTER 7

Renew Command

Calling the renew command will create and process a renewal order. If the domain has privacy, the renew command will also renew the privacy. You must have enough account balance to pay for the order.

Renew command args:

- domain - String of the domain you want to renew.
- duration - String/int of the duration in years you wish to renew for.

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.renew(domain="example.com", duration=1)
```

Response dict will contain:

- *result* - The result of the renew request.
- *more_info* - More information about the result. Only used when result is 'error'.
- *expiration_date* - Expiration date in unix time (milliseconds since midnight UTC of January 1, 1970). Only used when result is 'success'.

Renew command possible results:

```
from dynadotpy.client import RENEW_RESPONSES

RENEW_RESPONSES = {
    "success": "The domain was successfully renewed",
    "insufficient_funds": "Not enough account balance to process this renewal",
    "offline": "The central registry for this domain is currently offline",
    "error": "There was a syntax or registry error processing this request"
}
```

CHAPTER 8

Get Nameservers Command

Get the name servers for the specified domain.

Get name servers command args:

- domain - String of the domain.

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.get_nameservers(domain="example.com")
```

Response dict will contain:

- *result* - The result of the get name servers request.
- *ns[0-12]* - ns0-ns12 name servers. All 13 will always be returned. Any of them not set are simply blank strings.
- *more_info* - More information about the result. Only used when result is 'error'.

Get name servers command possible results:

```
from dynadotpy.client import GET_NAMESERVERS_RESPONSES

GET_NAMESERVERS_RESPONSES = {
    "success": "The nameservers were successfully return",
    "offline": "The central registry for this domain is currently offline",
    "error": "There was a syntax or registry error processing this request"
}
```

CHAPTER 9

Set Renew Option Command

Set the renewal options for a specific domain.

Set renew options command args:

- domain - String of the domain name to update; only one domain can be set per request.
- **option - String of the available renewal options. These are the only options allowed.**
  - *reset* - reset the domain's renew option to "no renew option"
  - *donot* - set the domain's renew option to "do not renew"
  - *auto* - set the domain's renew option to "auto-renew"

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.set_renew_option(domain="example.com", option="donot")
```

Response dict will contain:

- *result* - The result of the set renew options request.
- *more_info* - More information about the result. Only used when result is 'error'.

Set renew options command possible results:

```
from dynadotpy.client import SET_RENEW_RESPONSES

SET_RENEW_RESPONSES = {
    "success": "The renew options were successfully set.",
    "error": "There was a syntax error processing this request."
}
```

CHAPTER 10

Set Folder Command

Move the specified domain into a folder.

Warning: The folder you specify must already be created by using the Dynadot control panel.

Warning: Folder names **are** case sensitive. Test != test.

Set folder command args:

- domain - String of the domain name to update; only one domain can be set per request.
- folder - String of the folder name that you want to move your domain in.

```
from dynadotpy.client import Dynadot

dyn = Dynadot(api_key="<api_key>")
result = dyn.set_folder(domain="example.com", folder="test")
```

Response dict will contain:

- *result* - The result of the set renew options request.
- *more_info* - More information about the result. Only used when result is 'error'.

Set folder command possible results:

```
from dynadotpy.client import SET_FOLDER_RESPONSES

SET_FOLDER_RESPONSES = {
    "success": "The folder were successfully set",
    "error": "There was a syntax error processing this request"
}
```

CHAPTER 11

Indices and tables

- *genindex*
- *modindex*
- *search*