pyatomiadns Documentation

Release 1.4

Jochen Maes

February 17, 2014

Contents

This is the main and official documentation for pyatomiadns. pyatomiadns is still in BETA, test before use!

My name is Jochen Maes and you can contact me @ firstname dot sejo dash it dot be, I'm also online on freenode, oftc as sejo. I wrote this to be able to populate my dns servers with ansible.

The ansible module can be found under pyatomiadns/ansible/atomiadns.py.

You can find the code of this project here: https://github.com/sejo/pyatomiadns

One should also investigate https://github.com/sejo/django_atomiadns for a sane webapp.

CHAPTER 1

Thanks

For the release of the 1.3 release I would like to thank Amplidata NV for the time I was allowed to spend on creating and testing the 1.3 branch.

CHAPTER 2

How to test

On github in the vagrant folder you will see a Vagrant file and a crude ansible playbook. This playbook shows you how to install and configure atomiadns and runs some basic ansible commands for atomiadns.

I use the standard precise64 box:

```
vagrant box add precise64 http://files.vagrantup.com/precise64.box
```

Run above command to get the correct box.

CHAPTER 3

How to generate the api client file

I created a yaml file with most of the API calls that the Atomiadns json API supports, you can adapt however you want it. Each time to change it you should rebuild the pyatomiadns.client module. This is how to do it :

```
source venv/bin/activate #if your virtualenv is in venv
cd pyatomiadns/client_builder
python builder.py api.yaml
```

Running the previous commands should have generated a new client.py file.

Contents:

3.1 Generated API

class pyatomiadns.client.**AtomiaClient**(*url*, *email*, *password*)

The AtomiaClient plugs in on the json API from atomiadns.

AddAccount(*email*, *password_soap*)

Adds an account for soap

Parameters

- **email** – *str* email used as login
- **paswword_soap** – *str* password for the user

AddDNSSECKey(*algorithm*, *keysize*, *keytype*, *activated*)

Adds a DNSSEC key to the database. default algorithm: RSASHA256 for KSK use keysize 2048, for ZSK use 1024

Parameters

- **algorithm** – *str* defaults to RSASHA256
- **keysize** – *int* size of key in bit (1024, 2048,...)
- **keytype** – *str* KSK or ZSK
- **activated** – *str* yes or no

AddDnsRecords(*zone*, *records*)

Adds a list of records to a zone.

A record dict is the following: {

"ttl": "3600", "label" : "@", "class" : "IN", "type" : "A", "rdata" : "192.168.0.1"

}

Parameters

- **zone** – *str* the name of the zone
- **records** – *list* list of dicts containing the records'

AddNameserver(*nameserver*, *nameservergroup*)
Add a nameserver as a subscriber of changes to the data set in this server.

Parameters

- **nameserver** – *str* the servername to add as a subscriber
- **nameservergroup** – *str* the nameserver group that this nameserver should subscribe to changes for

AddNameserverGroup(*groupname*)
Add a nameserver group.

Parameters **groupname** – *str* name of the group you wish to add

AddZone(*zonename*, *zonettl*, *mname*, *rname*, *refresh*, *retry*, *expire*, *minimum*, *nameservers*, *nameservergroup*)
Add a zone to the Atomia DNS master database.

Parameters

- **zonename** – *str* the name of the zone
- **zonettl** – *int* the ttl of the SOA-record and the NS-records
- **mname** – *str* the SOA mname field
- **rname** – *str* the SOA rname field
- **refresh** – *int* the SOA refresh field
- **retry** – *int* the SOA retry field
- **expire** – *int* the SOA expire field
- **minimum** – *int* the SOA minimum field
- **nameservers** – *str* a string of the hostnames of the nameservers for the zone comma separated within brackets (["dns1.example.org","dns2.example.org"])
- **nameservergroup** – *str* the nameserver group that should host the zone

DeleteAccount(*email*)
Removes a soap account

Parameters **email** – *str* email of the account

DeleteDnsRecords(*zone*, *records*)
Removes the given records.

One should only provide the labels in a following format

'[{"label": "www"}, {"label": "bleh"}]'

Parameters

- **zone** – *str* the name of the zone
- **records** – *list* of *dict* containing only the label

DeleteNameServerGroup(*groupname*)
Delete a nameserver group.

Parameters **groupname** – *str* name of the group you wish to delete

DeleteNameserver(*nameserver*)
Remove a nameserver as a subscriber of changes to the data set in this server.

Parameters **nameserver** – *str* the servername to remove as a subscriber

DeleteZone(*zone*)
Deletes a zone from the database.

Parameters **zone** – *str* the name of the zone

EditAccount(*email*, *password*)
Allows one to change the password of ones account

Parameters

- **email** – *str* email of the account to change
- **password** – *str* new password

EditDnsRecords(*zone*, *records*)
Edits the records for a certain label

Parameters

- **zone** – *str* zone name
- **records** – *list* for format see AddDnsRecords

EditZone(*zonename*, *zonettl*, *mname*, *rname*, *refresh*, *retry*, *expire*, *minimum*, *nameservers*, *nameservergroup*)
Edits a zone. This is only for completeness, and could be done by editing the SOA and NS-records directly as well.

Parameters

- **zonename** – *str* the name of the zone
- **zonettl** – *int* the ttl of the SOA-record and the NS-records
- **mname** – *str* the SOA mname field
- **rname** – *str* the SOA rname field
- **refresh** – *int* the SOA refresh field
- **retry** – *int* the SOA retry field
- **expire** – *int* the SOA expire field
- **minimum** – *int* the SOA minimum field
- **nameservers** – *str* a string of the hostnames of the nameservers for the zone comma separated within brackets (["dns1.example.org","dns2.example.org"])
- **nameservergroup** – *str* the nameserver group that should host the zone

FindZones(*email*, *pattern*, *count*, *offset*)
Finds zones according to the pattern

Parameters

- **email** – *str* email of the account
- **pattern** – *str* the pattern to search for with SQL LIKE semantics

- **count** – *int* the max number of zones to return
- **offset** – *int* the offset of the first zone to return

GetAllZones()
GetAllZones returns all the zone names that are defined. This is an ADMIN only method

GetDnsRecords(*zone*, *label*)
GetRecord will fetch full record information for zone and label given

GetLabels(*zone*)
GetLabels returns all the labels for a certain zone.

Parameters **zone** – *str* zone (example.org, sejo-it.be,...)

GetNameserver(*nameserver*)
Gets the group name that a nameserver is configured as a subscriber for.

Parameters **nameserver** – *str* the servername to get information for

GetZone(*zone*)
GetZone returns the complete zone info with all records

Parameters **zone** – *str* zone (example.org, sejo-it.be,...)

GetZoneBinary(*zone*)
Gets the binary data for the zone, this can be used to restore zones

Parameters **zone** – *str* zone name

Noop()
Noop command used to authenticate

ReloadAllZones()
Mark all zones in the database as changed.

RestoreZoneBinary(*zone*, *nameservergroup*, *data*)
Restores the zone

Parameters

- **zone** – *str* zone name
- **nameservergroup** – *str* name of the nameserver group
- **data** – *str* data to restore

CHAPTER 4

Indices and tables

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

p