Introduction [1]

pydnsshim is an open-source client library written in python for the DNSSHIM server, which means it is intended to aid python developers in writing client applications that interact with the DNSSHIM server.
The protocol for communicating with DNSSHIM is XML-based, and what *pydnsshim* implements is a simple framework for generating the XML queries using the cheetah templating engine, communicating through the network and parsing the XML responses. Along with the *pydnsshim* library, an example shell-like client named *dnssh* comes in the package. It covers all the commands supported by DNSSHIM and works very well for testing and automating the task of running several operations in batch mode.

Requirements

- python 2.5
- cheetah 2.0.1 (template library)

Installation

To build and install pydnsshim, as well as the dnssh client simply run (as root):

```
$ python setup.py install
```

Pydnsshim basic usage

The following examples show the basic usage of the pydnsshim library of how to create requests, send them to the DNSSHIM server and get the response.

Login

First, in order to start using DNSSHIM, one must logged in, which can be accomplished as follows:

```
>>> from pydnsshim.LoginRequest import LoginRequest
>>> from pydnsshim.Transport import Transport
>>> from pydnsshim.LoginResponse import LoginResponse
>>> request = LoginRequest()
>>> request.username = 'username'
>>> request.password = 'password'
>>> requestXml = request.toXml()
```

[1] Version: *Rev* : 1365

```
>>> print requestXml

<?xml version="1.0" encoding="UTF-8"?>
<dnsshim version="1.0">
<request>
<login>
<username>username</username>
<password>0e457066c91fff7265565a4b26662058120af9dd</password>
</login>
</request>
</dnsshim>

>>> server = 'localhost'
>>> port = 9999
>>> transport = Transport(server, port)
>>> responseXml = transport.sendRequest(requestXml)
>>> response = LoginResponse()
>>> response.parseXml(responseXml)
>>> print response

Response status: 1
Message:

SessionID: 214662718

>>> sessionId = response.sessionId
```

Creating a new zone

Once logged, you can now start creating and managing zones. In order to create a new zone, the basic steps are:

```
>>> from pydnsshim.NewZoneRequest import NewZoneRequest
>>> from pydnsshim.NewZoneResponse import NewZoneResponse
>>> from pydnsshim.DnskeyAlgorithm import DnskeyAlgorithm
>>> request = NewZoneRequest()
>>> request.zone = 'example.com.br'
>>> request.ttl = 86400  # default
>>> request.mname = 'ns1.example.com.br'
>>> request.rname = 'hostmaster.example.com.br'
>>> request.serial = 1  # default
>>> request.refresh = 86400  # default
>>> request.retry = 900  # default
```

```
>>> request.expire = 604800  # default
>>> request.minimum = 900  # default
>>> request.signed = True  # default
>>> request.keySize = 1024  # default
>>> request.algorithm = DnskeyAlgorithm.RSASHA1 # default
>>> request.expirationPeriod = 2592000 # default
>>> requestXml = request.toXml()
>>> print requestXml

<?xml version="1.0" encoding="UTF-8"?>
<dnsshim version="1.0">
<request>
<newZone>
<sessionId>214662718</sessionId>
<zone>example.com.br</zone>
<dnssec>1</dnssec>
<soa>
<ttl>86400</ttl>
<mname>ns1.example.com.br</mname>
<rname>hostmaster.example.com.br</rname>
<serial>1</serial>
<refresh>86400</refresh>
<retry>900</retry>
<expire>604800</expire>
<minimum>900</minimum>
</soa>
<key>
<size>1024</size>
<algorithm>5</algorithm>
<expirationPeriod>2592000</expirationPeriod>
</key>
</newZone>
</request>
</dnsshim>

>>> responseXml = transport.sendRequest(requestXml)
>>> response = NewZoneResponse()
>>> response.parseXml(responseXml)
>>> print response

Response status: 1
Message:

NewZone:
```

```
Zone: example.com.br
Key Name: 20090505155650-ZSK-257-61669
Keytag: 61669
DS digest: e55b800be93839604df616a8a5da9fd295e7ce7a
Digest Type: SHA1
```

Listing the zones available

```
>>> from pydnsshim.ListZonesRequest import ListZonesRequest
>>> from pydnsshim.ListZonesResponse import ListZonesResponse
>>> request = ListZonesRequest()
>>> request.sessionId = sessionId
>>> requestXml = request.toXml()
>>> print requestXml

<?xml version="1.0" encoding="UTF-8"?>
<dnsshim version="1.0">
<request>
<listZones>
<sessionId>214662718</sessionId>
</listZones>
</request>
</dnsshim>

>>> responseXml = transport.sendRequest(requestXml)
>>> response = ListZonesResponse()
>>> response.parseXml(responseXml)
>>> print response
Response status: 1
Message:

Zones:
example.com.br
```

Adding a RR

```
>>> from pydnsshim.RrRequest import RrRequest
>>> from pydnsshim.RrType import RrType
>>> from pydnsshim.DnsClass import DnsClass
>>> from pydnsshim.RrOperation import RrOperation
>>> from pydnsshim.Response import Response
>>> request = RrRequest()
```

```
>>> request.sessionId = sessionId
>>> request.op = RrOperation.ADD
>>> request.zone = 'example.com.br'
>>> request.ownername = ''  # Apex
>>> request.ttl = 86400  # default
>>> request.type = RrType('NS')
>>> request.dnsClass = DnsClass.IN  # default
>>> request.rdata = 'ns1.example.com.br'
>>> requestXml = request.toXml()
>>> print requestXml

<?xml version="1.0" encoding="UTF-8"?>
<dnsshim version="1.0">
<request>
<addRr>
<sessionId>214662718</sessionId>
<zone>example.com.br</zone>
<rr>
<ownername></ownername>
<ttl>86400</ttl>
<type>NS</type>
<dnsClass>1</dnsClass>
<rdata>ns1.example.com.br</rdata>
</rr>
</addRr>
</request>
</dnsshim>

>>> responseXml = transport.sendRequest(requestXml)
>>> response = Response()
>>> response.parseXml(responseXml)
>>> print response

Response status: 1
Message:
```

Printing the contents of zone

```
>>> from pydnsshim.PrintZoneRequest import PrintZoneRequest
>>> from pydnsshim.PrintZoneResponse import PrintZoneResponse
>>> request = PrintZoneRequest()
>>> request.sessionId = sessionId
>>> request.zone = 'example.com.br'
>>> requestXml = request.toXml()
```

```
>>> print requestXml

<?xml version="1.0" encoding="UTF-8"?>
<dnsshim version="1.0">
<request>
<printZone>
<sessionId>214662718</sessionId>
<zone>example.com.br</zone>
</printZone>
</request>
</dnsshim>

>>> responseXml = transport.sendRequest(requestXml)
>>> response = PrintZoneResponse()
>>> response.parseXml(responseXml)
>>> print response
Response status: 1
Message:

Zone: example.com.br
SOA ns1.example.com.br hostmaster.example.com.br 1 86400 900 604800 900
NS ns1.example.com.br.

DS Info:
61669 e55b800be93839604df616a8a5da9fd295e7ce7a
```

Dnssh Usage

In order to start the dnssh client, simply type at the command-line:
dnssh.py [-h] [-s server_IP] [-p port] [-u username -P password]

-h Print usage instructions

-s *server* Server's IP address

-p *port* Server's Port

-u *username* Automatically try to login as *username* (Password required)

-P *password* Password used to login

After dnssh is running, you can start issuing commands to the DNSSHIM server.

Available commands

The following commands are available:

login

Logs into the server using username and password.

```
dnssh> login -u username
```

-u *username* Username.

In the response, it returns the session ID for the user's current session.

logout

Finishes the current session.

```
dnssh> logout
```

hello

Simply keeps the session alive, avoiding a timeout.

```
dnssh> hello
```

add-user

Creates a new user.

```
dnssh> add-user -u username
```

-u *username* Username of the new user.

change-password

Changes the password for user *username*.

```
dnssh> change-password -u username
```

-u *username* Username whose password is being changed.

new-zone

Creates a new zone. By the default the new zone is already signed using the DNSSEC protocol extension, so a new DNSKEY is generated automatically.

```
dnssh> new-zone -z zonename [--ttl=ttl --mname=mname --rname=rname
    --serial=serial --refresh=refresh --retry=retry --expire=expire
    --minimum=minimum --key-size=key_size --rrsig-validity=signature_validity
    --no-dnssec ]
```

-z *zonename*	Name of the zone to be created.
–ttl=*ttl*	TTL of the SOA record (default is 86400).
–mname=*mname*	MNAME parameter of the SOA record (default is "ns1."*zonename*).
–rname=*rname*	RNAME parameter of the SOA record (default is "hostmaster."*zonename*).
–serial=*serial*	SERIAL version number of the new zone (default is 1).
–refresh=*refresh*	REFRESH parameter of the SOA record (default is 86400).
–retry=*retry*	RETRY parameter of the SOA record (default is 900).
–expire=*expire*	EXPIRE parameter of the SOA record (default is 604800).
–minimum=*minimum*	MINIMUM parameter of the SOA record (default is 900).
–key-size=*key_size*	Key size of the zone's DNSKEY (default is 1024).
–rrsig-validity=*signature_validity*	Validity of the zone's signatures (default is 1 month).
–no-dnssec	Create a zone without DNSSEC.

In the response, it returns the zone name and, if the zone is signed with DNSSEC, some data related to the newly generated DNSKEY, such as the key name, keytag and the DS digest that is to be included in the parent zone.

remove-zone

Removes a zone from the DNSSHIM server.

```
dnssh> remove-zone -z zonename
```

-z *zonename* The zone to be removed.

In the response, it returns the name of the removed zone.

print-zone

Prints the records of a zone

```
dnssh> print-zone -z zonename
```

-z *zonename* The zone to be printed.

In the response, it returns the name of the zone, followed by a list of all of its records and the DS information that is to be included in the parent zone.

pub-zone

Publishes the zone to the configured slave servers

```
dnssh> pub-zone -z zonename [-s <serial> -i|--incremental -f|--full ]
```

-z *zonename*	The zone to be published.
-s *serial*	The new version number of the zone (Must be greater than the current one).
-i|–incremental	Perform an incremental publication (IXFR).
-f|–full	Perform a full publication (AXFR).

In the response, it returns the name of the zone and the current serial version.

list-zones

Lists all the zones managed by the current user

```
dnssh> list-zones
```

In the response, it returns a list containing all the zone names managed by the current user.

add-rr and remove-rr

Adds or Removes a Resource Record from the zone

```
dnssh> add-rr|remove-rr -z zonename -o ownername -t type -r ''rdata'' [--ttl=ttl]
```

-z *zonename*	The zone to be published.
-o *ownername*	Ownername of the record being added or removed (default is the apex).
-t *type*	Type of the record.
-r "*rdata*"	RDATA of the record in presentation format.
–ttl=*ttl*	TTL of the record to add or remove.

import-zone

Imports a zone from an authoritative server using AXFR.

```
dnssh> import-zone -z zonename -s server_IP [-p port]
```

-z *zonename*	Name of the zone to import.
-s *server_IP*	IP of the authoritative server to import from.
-p *port*	Port number on the authoritative server (default is 53).

In the response, it returns the name of the newly imported zone.

zone-version

Prints the current version (SOA serial) of the zone.

```
dnssh> zone-version -z zonename
```

-z *zonename*	Zone whose version will be printed.

In the response, it returns the name of the zone and its current serial version.

set-expiration-period

Sets the signatures's validity period of zone.

```
dnssh> set-expiration-period -z zonename [--rrsig-validity=signature_validity]
```

-z *zonename*	Name of the zone.
–rrsig-validity=*signature_validity*	Validity of the zone's signatures (default is 1 month).

In the response, it returns the name of the zone and the period that was actually set. That may differ from the value sent in the request because the DNSSHIM server has some restrictions regarding this parameters minimum and maximum value.

add-zone-user and remove-zone-user

Adds or removes an user as one of the administrators of a zone.

```
dnssh> add-zone-user|remove-zone-user -z zonename -u username
```

-z *zonename*	Name of the zone.
-u *username*	Username of an existing user.

new-key

Generates a new key for a specified zone.

```
dnssh> new-key -z zonename [--key-size=key_size --ksk|--zsk --flags=flags
    --status=key_status]
```

-z *zonename*	Name of the zone.
–key-size=*key_size*	Size of the new key (default is 1024).
–ksk	Make the new key a Key Signing Key.
–zsk	Make the new key a Zone Signing Key.
–flags=*flags*	The flags field of the new key (Either 256 or 257. Default is 257).
–status=*key_status*	Status of the new key (Either SIGN, PUBLISH or NONE. Default is SIGN).

In the response, it returns the name of the zone, followed by the name of the newly generated key, its keytag and the related DS information that is to be included in parent zone.

import-key

Imports an existing key from a BIND private key file into a zone.

```
dnssh> import-key -z zonename -k key_file [--ksk|--zsk --flags=flags
    --status=key_status]
```

-z *zonename*	Name of the zone.
-k *key_file*	File, as generated by BIND, containing the private key.
–ksk	Make the new key a Key Signing Key.
–zsk	Make the new key a Zone Signing Key.
–flags=*flags*	The flags field of the new key (Either 256 or 257. Default is 257).
–status=*key_status*	Status of the new key (Either SIGN, PUBLISH or NONE. Default is SIGN).

In the response, it returns the name of the zone, followed by the name of the newly generated key, its keytag and the related DS information that is to be included in parent zone.

list-keys

List all the keys associated to a zone.

```
dnssh> list-keys -z zonename
```

-z *zonename* Name of the zone.

In the response, it returns the name of the zone, followed by a list containing all the keys grouped by the status of the key.

change-key-status

Changes the status of a specified key.

```
dnssh> change-key-status -z zonename -k key_name -o|--old-status=old_status
    -n|--new-status=new_status
```

-z *zonename*	Name of the zone.
-k *key_name*	Name of the key to be modified.
-o|–old-status *old_status*	Current status of the key.
-n|–new-status *new_status*	New status of the key.

In the response, it returns the keytag and the related DS information that is to be included in the parent zone.

new-slavegroup and remove-slavegroup

Creates and removes a group of slave nameservers, aka slavegroup.

```
dnssh> new-slavegroup|remove-slavegroup -g slavegroup_name
```

-g *slavegroup_name* Name of the slavegroup to be created or removed.

list-slavegroups

List all the slavegroups.

```
dnssh> list-slavegroups
```

In the response, it returns a list containing all the slavegroup names available.

assign-slavegroup and unassign-slavegroup

Assigns and unassigns a slavegroup to and from a zone.

```
dnssh> assign-slavegroup|unassign-slavegroup -z zonename -g slavegroup_name
```

-z *zonename*	Name of the zone.
-g *slavegroup_name*	Name of the slavegroup.

list-zones-by-slavegroup

List all the zones which have a specific slavegroup assigned to it.

```
dnssh> list-zones-by-slavegroup -g slavegroup_name
```

-g *slavegroup_name* Name of the slavegroup.

In the response, it returns the name of the slavegroup and a list containing all the zone names associated to it.

add-slave and remove-slave

Adds and removes a slave nameserver to and from a slavegroup.

```
dnssh> add-slave|remove-slave -g slavegroup_name -s slave_IP [-p port]
```

-g *slavegroup_name* Name of the slavegroup.
-s *slave_IP* IP address of the slave nameserver.
-p *port* Port number of the slave nameserver (default is 53).

list-slaves

List all the slave nameservers assigned to a zone.

```
dnssh> list-slaves -z zonename
```

-z *zonename* Name of the zone.

In the response, it returns the name of the zone and a list containing all the slavegroups and slaves assigned to it.

print-slavegroup

Print all the slave nameservers of a slavegroup.

```
dnssh> print-slavegroup -g slavegroup_name
```

-g *slavegroup_name* Name of the slavegroup.

In the response, it returns the name of the slavegroup and a list containing all the slaves in it.

new-tsig-key

Creates a new TSIG key.

```
dnssh> new-tsig-key -s server_IP -k key_name
```

-s *server_IP* IP address of the server associated to the TSIG key.
-k *key_name* Name of the new TSIG key.

In the response, it returns the name key, the IP address of the server associated to it and its secret.

remove-tsig-key

Removes an existing TSIG key.

```
dnssh> remove-tsig-key -s server_IP -k key_name
```

-s *server_IP* IP address of the server associated to the TSIG key.
-k *key_name* Name of the TSIG key to be removed.

list-tsig-keys

List all TSIG keys associated to a server.

```
dnssh> list-tsig-keys -s server_IP
```

-s *server_IP* IP address of the server.

In the response, it returns the IP address of the server and a list with all of its TSIG key names and secrets.